```
=> d his full

     (FILE 'HOME' ENTERED AT 12:15:52 ON 24 SEP 2001)

     FILE 'USPATFULL' ENTERED AT 12:15:56 ON 24 SEP 2001
L1        105706 SEA (NOTIF? OR INFORM### OR UPDAT? OR ALERT? OR REPORT? OR
                 INDICAT?) (5A) (PROGRAM# OR ROUTINE# OR APPLICATION# OR
                 SOFTWARE# OR CLIENT# OR TASK# OR PROCESS OR PROCESSES)
L2         16766 SEA L1 (5A) (CONFIG? OR RECONFIG? OR CHANG? OR MODIF? OR
                 ALTER? OR VAR? OR CONNECT? OR DISCONNECT? OR COUPL? OR
                 DECOUPL? OR PLUG? OR UNPLUG? OR REMOV? OR WITHDRAW# OR
                 INSERT? OR ADD? OR EXTRACT? OR EJECT?)
                 D PN,TI,IN,NCL,AB,HIT 1-3
L3          1599 SEA (((NOTIF?/AB,CLM OR INFORM###/AB,CLM OR UPDAT?/AB,CLM OR
                 ALERT?/AB,CLM OR REPORT?/AB,CLM OR INDICAT?/AB,CLM) (5A)
                 (PROGRAM#/AB,CLM OR ROUTINE#/AB,CLM OR APPLICATION#/AB,CLM
                 OR SOFTWARE#/AB,CLM OR CLIENT#/AB,CLM OR TASK#/AB,CLM OR
                 PROCESS/AB,CLM OR PROCESSES/AB,CLM)) (5A) (CONFIG?/AB,CLM OR
                 RECONFIG?/AB,CLM OR CHANG?/AB,CLM OR MODIF?/AB,CLM OR
                 ALTER?/AB,CLM OR VAR?/AB,CLM OR CONNECT?/AB,CLM OR DISCONNECT
                 ?/AB,CLM OR COUPL?/AB,CLM OR DECOUPL?/AB,CLM OR PLUG?/AB,CLM
                 OR UNPLUG?/AB,CLM OR REMOV?/AB,CLM OR WITHDRAW#/AB,CLM OR
                 INSERT?/AB,CLM OR ADD?/AB,CLM OR EXTRACT?/AB,CLM OR EJECT?/AB
                 ,CLM))
L4           121 SEA (((NOTIF?/AB OR INFORM###/AB OR UPDAT?/AB OR ALERT?/AB
                 OR REPORT?/AB OR INDICAT?/AB) (5A) (PROGRAM#/AB OR ROUTINE#/A
                 B OR APPLICATION#/AB OR SOFTWARE#/AB OR CLIENT#/AB OR
                 TASK#/AB OR PROCESS/AB OR PROCESSES/AB)) (5A) (CONFIG?/AB OR
                 RECONFIG?/AB OR CHANG?/AB OR MODIF?/AB OR ALTER?/AB OR
                 VAR?/AB OR CONNECT?/AB OR DISCONNECT?/AB OR COUPL?/AB OR
                 DECOUPL?/AB OR PLUG?/AB OR UNPLUG?/AB OR REMOV?/AB OR
                 WITHDRAW#/AB OR INSERT?/AB OR ADD?/AB OR EXTRACT?/AB OR
                 EJECT?/AB)) AND (((NOTIF?/CLM OR INFORM###/CLM OR UPDAT?/CLM
                 OR ALERT?/CLM OR REPORT?/CLM OR INDICAT?/CLM) (5A) (PROGRAM#/
                 CLM OR ROUTINE#/CLM OR APPLICATION#/CLM OR SOFTWARE#/CLM OR
                 CLIENT#/CLM OR TASK#/CLM OR PROCESS/CLM OR PROCESSES/CLM))
                 (5A) (CONFIG?/CLM OR RECONFIG?/CLM OR CHANG?/CLM OR MODIF?/CL
                 M OR ALTER?/CLM OR VAR?/CLM OR CONNECT?/CLM OR DISCONNECT?/CL
                 M OR COUPL?/CLM OR DECOUPL?/CLM OR PLUG?/CLM OR UNPLUG?/CLM
                 OR REMOV?/CLM OR WITHDRAW#/CLM OR INSERT?/CLM OR ADD?/CLM OR
                 EXTRACT?/CLM OR EJECT?/CLM))
                 D PN,TI,IN,NCL,AB 1-3
L5            90 SEA L4 AND 7##/NCL
                 SET LINELENGTH 78
                 D PN,TI,IN,NCL,AB 1-
                 SET HIGH OFF
L6            62 SEA L5 AND REQUEST###
L7         22478 SEA (NOTIF? OR INFORM### OR UPDAT? OR ALERT? OR REPORT? OR
                 INDICAT?) (5A) REQUEST###
L8          9630 SEA L7 (P) (PROGRAM# OR ROUTINE# OR APPLICATION# OR SOFTWARE#
                  OR CLIENT# OR TASK# OR PROCESS OR PROCESSES)
L9         10000 SEA L1 (P) REQUEST###
L10        14631 SEA L8 OR L9
L11           44 SEA L5 AND L10
                 D PN,TI,IN,NCL,AB,HIT 1-
                 SET HIGH ON
L12        22478 SEA (NOTIF? OR INFORM### OR UPDAT? OR ALERT? OR REPORT? OR
                 INDICAT?) (5A) REQUEST###
```

```
L13          9630 SEA L12 (P) (PROGRAM# OR ROUTINE# OR APPLICATION# OR
                  SOFTWARE# OR CLIENT# OR TASK# OR PROCESS OR PROCESSES)
L14         10000 SEA L1 (P) REQUEST###
L15         14631 SEA L13 OR L14
                  SET HIGH OFF
L16          1285 SEA (((NOTIF?/CLM OR INFORM###/CLM OR UPDAT?/CLM OR ALERT?/CL
                  M OR REPORT?/CLM OR INDICAT?/CLM) (5A) (PROGRAM#/CLM OR
                  ROUTINE#/CLM OR APPLICATION#/CLM OR SOFTWARE#/CLM OR
                  CLIENT#/CLM OR TASK#/CLM OR PROCESS/CLM OR PROCESSES/CLM))
                  (5A) (CONFIG?/CLM OR RECONFIG?/CLM OR CHANG?/CLM OR MODIF?/CL
                  M OR ALTER?/CLM OR VAR?/CLM OR CONNECT?/CLM OR DISCONNECT?/CL
                  M OR COUPL?/CLM OR DECOUPL?/CLM OR PLUG?/CLM OR UNPLUG?/CLM
                  OR REMOV?/CLM OR WITHDRAW#/CLM OR INSERT?/CLM OR ADD?/CLM OR
                  EXTRACT?/CLM OR EJECT?/CLM))
                  SET HIGH ON
L17           435 SEA (((NOTIF?/AB OR INFORM###/AB OR UPDAT?/AB OR ALERT?/AB
                  OR REPORT?/AB OR INDICAT?/AB) (5A) (PROGRAM#/AB OR ROUTINE#/A
                  B OR APPLICATION#/AB OR SOFTWARE#/AB OR CLIENT#/AB OR
                  TASK#/AB OR PROCESS/AB OR PROCESSES/AB)) (5A) (CONFIG?/AB OR
                  RECONFIG?/AB OR CHANG?/AB OR MODIF?/AB OR ALTER?/AB OR
                  VAR?/AB OR CONNECT?/AB OR DISCONNECT?/AB OR COUPL?/AB OR
                  DECOUPL?/AB OR PLUG?/AB OR UNPLUG?/AB OR REMOV?/AB OR
                  WITHDRAW#/AB OR INSERT?/AB OR ADD?/AB OR EXTRACT?/AB OR
                  EJECT?/AB))
L18           121 SEA L16 AND L17
L19            45 SEA L18 AND L15
L20            44 SEA L19 AND 7##/NCL
                  D PN,TI,IN,NCL,AB,HIT 1-
L21          8849 SEA L1 (5A) (CONNECT? OR DISCONNECT? OR COUPL? OR DECOUPL?
                  OR PLUG? OR UNPLUG? OR REMOV? OR WITHDRAW# OR INSERT? OR
                  ADD? OR EXTRACT? OR EJECT?)
L22            33 SEA L20 AND L21
                  D PN,TI,IN,NCL,AB,HIT 1-
```

```
=> d his full

     (FILE 'HOME' ENTERED AT 13:33:55 ON 24 SEP 2001)

     FILE 'USPATFULL' ENTERED AT 13:33:59 ON 24 SEP 2001
L1        156221 SEA (NOTIF? OR INFORM? OR UPDAT? OR ALERT? OR REPORT? OR
                 INDICAT?) (5A) (PROGRAM# OR ROUTINE# OR APPLICATION# OR
                 SOFTWARE# OR CLIENT# OR TASK# OR PROCESS OR PROCESSES)
L2         70697 SEA L1 (P) (CONNECT? OR DISCONNECT? OR COUPL? OR DECOUPL? OR
                 PLUG? OR UNPLUG? OR REMOV? OR WITHDRAW# OR INSERT? OR ADD?
                 OR EXTRACT? OR EJECT?)
L3         35803 SEA REQUEST### (5A) (NOTIF? OR INFORM? OR UPDAT? OR ALERT?
                 OR REPORT? OR INDICAT?)
L4         15052 SEA L2 AND L3
L5         10476 SEA L4 AND 7##/NCL
                 SET HIGH OFF
L6        339566 SEA (NOTIF? OR INFORM? OR UPDAT? OR ALERT? OR REPORT? OR
                 INDICAT?)/CLM
L7       1763174 SEA (CONNECT? OR DISCONNECT? OR COUPL? OR DECOUPL? OR PLUG?
                 OR UNPLUG? OR REMOV? OR WITHDRAW# OR INSERT? OR ADD? OR
                 EXTRACT? OR EJECT?)/CLM
L8        253947 SEA L6 AND L7
                 SET HIGH ON
L9          7488 SEA L5 AND L8
                 SET HIGH OFF
L10         7417 SEA (((NOTIF?/CLM OR INFORM?/CLM OR UPDAT?/CLM OR ALERT?/CLM
                 OR REPORT?/CLM OR INDICAT?/CLM) (5A) (PROGRAM#/CLM OR
                 ROUTINE#/CLM OR APPLICATION#/CLM OR SOFTWARE#/CLM OR
                 CLIENT#/CLM OR TASK#/CLM OR PROCESS/CLM OR PROCESSES/CLM))
                 (P) (CONNECT?/CLM OR DISCONNECT?/CLM OR COUPL?/CLM OR
                 DECOUPL?/CLM OR PLUG?/CLM OR UNPLUG?/CLM OR REMOV?/CLM OR
                 WITHDRAW#/CLM OR INSERT?/CLM OR ADD?/CLM OR EXTRACT?/CLM OR
                 EJECT?/CLM))
                 SET HIGH ON
L11         3076 SEA (((NOTIF?/AB OR INFORM?/AB OR UPDAT?/AB OR ALERT?/AB OR
                 REPORT?/AB OR INDICAT?/AB) (5A) (PROGRAM#/AB OR ROUTINE#/AB
                 OR APPLICATION#/AB OR SOFTWARE#/AB OR CLIENT#/AB OR TASK#/AB
                 OR PROCESS/AB OR PROCESSES/AB)) (P) (CONNECT?/AB OR DISCONNEC
                 T?/AB OR COUPL?/AB OR DECOUPL?/AB OR PLUG?/AB OR UNPLUG?/AB
                 OR REMOV?/AB OR WITHDRAW#/AB OR INSERT?/AB OR ADD?/AB OR
                 EXTRACT?/AB OR EJECT?/AB))
L12         1086 SEA L10 AND L11
L13          409 SEA L3 AND L12
L14          340 SEA L13 AND 7##/NCL
L15           65 SEA L14 AND (710/NCL OR 713/NCL)
                 D PN,TI,IN,NCL,AB,HIT 1-3
                 SET HIGH OFF
L16         7979 SEA (REQUEST###/CLM (5A) (NOTIF?/CLM OR INFORM?/CLM OR
                 UPDAT?/CLM OR ALERT?/CLM OR REPORT?/CLM OR INDICAT?/CLM))
                 SET HIGH ON
L17          158 SEA L14 AND L16
L18           31 SEA L17 AND (710/NCL OR 713/NCL)
                 SET LINELENGTH 78
                 D PN,TI,IN,NCL,AB,HIT 1-
L19          181 SEA L12 AND L16
L20           31 SEA L19 AND (710/NCL OR 713/NCL)
                 D PN,TI,IN,NCL,AB,HIT 1-3
                 D PN,TI,IN,NCL,AB 1-
```

L21 21 SEA L14 AND (CALLBACK OR CALL-BACK)

L22 20 S L21 NOT L20

D AN,TI,IN,NCL,AB,HIT 1-

L18 ANSWER 1 OF 76 USPATFULL
PI US 2002016867 A1 20020207
TI Cluster event service method and system
IN Kampe, Mark A., Los Angeles, CA, UNITED STATES
Herrmann, Frederic, Palo Alto, CA, UNITED STATES
Fernandez, Ludovic Christophe, San Francisco, CA, UNITED STATES
NCL NCLM: **709/318.000**
AB A network having a plurality of nodes is disclosed. The network includes an event channel adapted to transmit an event between a publisher node and a subscriber node within the network. The network also includes a filter to identify the event on the subscriber node. The network also includes an application on the subscriber node to receive the event according to the filter.
CLM What is claimed is:
44. A method for opening an event channel within a network, comprising: receiving a request from an application on a node for said event channel; **registering** a **callback** function from said **application**; and subscribing to an event for said application on said event channel.

49. A method for opening an event channel within a network, comprising: receiving a request from an application on a node for said event channel; **registering** a **callback** function from said **application**; and publishing an event from said application on said event channel.

88. A computer program product comprising a computer useable medium having computer readable code embodied therein for opening an event channel within a network, the computer program product adapted when run on a computer to effect steps, including: receiving a request from an application on a node for said event channel; **registering** a **callback** function from said **application**; and subscribing to an event for said application on said event channel.

89. A computer program product comprising a computer useable medium having computer readable code embodied therein for opening an event channel within a network, the computer program product adapted when run on a computer to effect steps including: receiving a request from an application on a node for said event channel; **registering** a **callback** function from said **application**; and publishing an event from said application on said event channel.

NCL NCLM: **709/318.000**

L18 ANSWER 4 OF 76 USPATFULL
PI US 2002002614 A1 20020103
TI Dynamic lookup service in distributed system
IN Murphy, Brian T., Milford, NH, UNITED STATES
Scheifler, Robert W., Somerville, MA, UNITED STATES
Pan, Zane, Lexington, MA, UNITED STATES
Waldo, James H., Dracut, MA, UNITED STATES
Wollrath, Ann M., Groton, MA, UNITED STATES
Arnold, Kenneth C.R.C., Lexington, MA, UNITED STATES
NCL NCLM: **709/226.000**
NCLS: **709/221.000**
AB An improved lookup service is provided that allows for the dynamic addition and deletion of services. This lookup service allows for the addition and deletion of services automatically, without user intervention. As a result, clients of the lookup service may continue using the lookup service and its associated services while the updates occur. Additionally, the lookup service provides a notification mechanism that can be used by clients to receive a notification when the lookup service is updated. By receiving such a notification, clients can avoid attempting to access a service that is no longer available and can make use of new services as soon as they are added to the lookup service.
CLM What is claimed is:
6. The method of claim 5 wherein the registering step includes the step of: **registering** a **callback routine** with the lookup service, and wherein the sending step includes the step of: invoking the **registered callback routine**.

7. The method of claim 6 wherein the registering step includes the step of: registering a parameter object with the lookup service, and wherein the invoking step includes the step of: invoking the **registered callback routine** passing the parameter object as a parameter to the callback routine.

68. The computer-readable medium of claim 64 wherein the method further includes the step of registering with the event mechanism of the lookup service, said registration resulting in a **request** that the lookup service **notify** the **client** lookup manager of a change to its associated network services.

NCL NCLM: **709/226.000**
NCLS: **709/221.000**|

L18 ANSWER 8 OF 76 USPATFULL
PI US 6314471 B1 20011106
TI Techniques for an interrupt free operating system
IN Alverson, Gail A., Seattle, WA, United States
Callahan, II, Charles David, Seattle, WA, United States
Coatney, Susan L., Federal Way, WA, United States
Kaplan, Laurence S., Seattle, WA, United States
Korry, Richard D., Seattle, WA, United States
NCL NCLM: **710/005.000**
NCLS: **710/006.000; 710/018.000;**
712/009.000; 712/215.000; 712/241.000
AB A method and system in a multithreaded processor for processing events without interrupt notifications. In one aspect of the present invention, an operating system creates a thread to execute on a stream of the processor. During execution of the thread, the thread executes a loop that determines whether an event has occurred and, in response to determining whether an event has occurred, assigns a different thread to process the event so that multiple events can be processed in parallel and so that interrupts are not needed to signal that the event has occurred. Another aspect of the present invention provides a method and system for processing asynchronously occurring events without interrupt notifications. To achieve this processing, a first thread is executed to generate a notification that the event has occurred upon receipt of the asynchronously occurring event. A second thread is also executed that loops determining whether a notification has been generated and, in response to determining that a notification has been generated, performing the processing necessary for the event.
CLM What is claimed is:
41. The computer system of claim 39 wherein device **drivers** invoke **callback routines registered** on behalf of user **programs**.

NCL NCLM: **710/005.000**
NCLS: **710/006.000; 710/018.000;**
712/009.000; 712/215.000; 712/241.000
|

L18 ANSWER 15 OF 76 USPATFULL
PI US 6256632 B1 20010703
TI Method and system for enabling dynamic cache structures in a networked environment
IN Fraenkel, Michael L., Raleigh, NC, United States
Nguyen, Binh Q., Cary, NC, United States
Singhal, Sandeep Kishan, Raleigh, NC, United States
NCL NCLM: **707/010.000**
NCLS: **707/003.000**
AB Methods, systems and computer program products are provided for managing shared data elements among a plurality of different client processes in a network environment. Shared data elements are associated with a Flow. A Flow is a logical stream of data that is only transmitted to a client process that explicitly subscribes for updates from the Flow. Update requests for the shared data elements are transmitted from client processes along the Flow so as to request the receipt of update notifications along the Flow. Update notifications are also transmitted about the shared data elements to the client **processes** along the Flow which have **requested** update **notifications**. Content of the shared data elements is, thereby, delivered to applications executing within said at least one client process which have requested updates of the shared data elements.
AB Methods, systems and computer program products are provided for managing shared data elements among a plurality of different client processes in a network environment. Shared data elements are associated with a Flow. A Flow is a logical stream of data that is only transmitted to a client process that explicitly subscribes for updates from the Flow. Update requests for the shared data elements are transmitted from client processes along the Flow so as to request the receipt of update notifications along the Flow. Update notifications are also transmitted about the shared data elements to the client **processes** along the Flow which have **requested** update **notifications**. Content of the shared data elements is, thereby, delivered to applications executing within said at least one client process which have requested updates of the shared data elements.
NCL NCLM: **707/010.000**
NCLS: **707/003.000**

L18 ANSWER 20 OF 76 USPATFULL
PI US 2001002473 A1 20010531
TI Dynamic lookup service in a distributed system
IN Waldo, James H., Dracut, MA, United States
Wollrath, Ann M., Groton, MA, United States
Scheifler, Robert W., Somerville, MA, United States
Arnold, Kenneth C.R.C., Lexington, MA, United States
NCL NCLM: **709/229.000**
NCLS: **709/220.000; 709/227.000;**
709/225.000
AB An improved lookup service is provided that allows for the dynamic addition and deletion of services. This lookup service allows for the addition and deletion of services automatically without user intervention. As a result, clients of the lookup service may continue using the lookup service and its associated services while the updates occur. Additionally, the lookup service provides a notification mechanism that can be used by clients to receive a notification when the lookup service is updated. By receiving such a notification, clients can avoid attempting to access a service that is no longer available and can make use of new services as soon as they are added to the lookup service.
CLM What is claimed is:
6. The method of claim 5 wherein the registering step includes the step of: **registering** a **callback routine** with the lookup service, and wherein the sending step includes the step of: invoking the **registered callback routine**.

7. The method of claim 6 wherein the registering step includes the step of: registering a parameter object with the lookup service, and wherein the invoking step includes the step of: invoking the **registered callback routine** passing the parameter object as a parameter to the callback routine.

NCL NCLM: **709/229.000**
NCLS: **709/220.000; 709/227.000;**
709/225.000|

L18 ANSWER 28 OF 76 USPATFULL
PI US 6185611 B1 20010206
TI Dynamic lookup service in a distributed system
IN Waldo, James H., Dracut, MA, United States
Wollrath, Ann M., Groton, MA, United States
Scheifler, Robert W., Somerville, MA, United States
Arnold, Kenneth C. R. C., Lexington, MA, United States
NCL NCLM: **709/221.000**
NCLS: **709/201.000; 709/223.000;**
709/224.000
AB An improved lookup service is provided that allows for the dynamic addition and deletion of services. This lookup service allows for the addition and deletion of services automatically, without user intervention. As a result, clients of the lookup service may continue using the lookup service and its associated services while the updates occur. Additionally, the lookup service provides a notification mechanism that can be used by clients to receive a notification when the lookup service is updated. By receiving such a notification, clients can avoid attempting to access a service that is no longer available and can make use of new services as soon as they are added to the lookup service.
CLM What is claimed is:
6. The method of claim 5 wherein the registering step includes the step of: **registering** a **callback routine** with the lookup service, and wherein the sending step includes the step of: invoking the **registered callback routine**.

7. The method of claim 6 wherein the registering step includes the step of: registering a parameter object with the lookup service, and wherein the invoking step includes the step of: invoking the **registered callback routine** passing the parameter object as a parameter to the callback routine.

NCL NCLM: **709/221.000**
NCLS: **709/201.000; 709/223.000;**
709/224.000|

L18 ANSWER 32 OF 76 USPATFULL
PI US 6101528 20000808
TI Method and apparatus for discovering server applications by a client application in a network of computer systems
IN Butt, Alan, Orem, UT, United States
NCL NCLM: **709/203.000**
NCLS: **709/235.000; 709/245.000**
AB In a network of computer systems, each node having client applications that perform server application discovery is provided with a client discovery service having a first collection of functions, including a first **registration** function for **registering** discovery **callback** procedures of **client applications**, and a send function for sending discovery packets on behalf of the client applications. Additionally, each node having server applications that are interested in being discovered by client applications is provided with a server discovery service having a second collection of functions, including a second registration function for registering server applications interested in being discovered, and a respond function for sending discovery response packets on behalf of the registered server applications. During operation, these functions facilitate the client applications in performing server application discovery in a manner that reduces network traffics.
AB In a network of computer systems, each node having client applications that perform server application discovery is provided with a client discovery service having a first collection of functions, including a first **registration** function for **registering** discovery **callback** procedures of **client applications**, and a send function for sending discovery packets on behalf of the client applications. Additionally, each node having server applications that are interested in being discovered by client applications is provided with a server discovery service having a second collection of functions, including a second registration function for registering server applications interested in being discovered, and a respond function for sending discovery response packets on behalf of the registered server applications. During operation, these functions facilitate the client applications in performing server application discovery in a manner that reduces network traffics.
CLM What is claimed is:
11. A network of computer systems comprising: (a) a client system having a **client** discovery service including a first **registration** service to **register** a **notification** mechanism of a **client application** of the client system, the client system further having a send discovery packet service to send discovery packets to a server system interconnected with the client system on demand, on behalf and at the request of the registered client application, wherein the client discovery service facilitates discovery of whether a server application that is of interest to the registered client application is available on the server system independent of the server application's knowledge of the client application; and (b) the server system, including a server discovery service having a second **registration** service to **register** a **notification** mechanism of the server **application**, wherein the registered server application is available to interested client applications of the client system, including client applications that the server application has no knowledge of, and a

respond service to listen for discovery packets from the client discovery service(s) of the client system, and to reply to the detected discovery packets with discovery response packets on behalf of the registered server application, as appropriate.

23. In a network of computer systems, a computer implemented method for discovering a server application by a client application independent of the server application's knowledge, if any, of the client application, the method comprising: a) the **client application** independently **registering** its **notification** mechanism with an independent **client** discovery service; b) the independent client discovery service sending a discovery packet on demand, on behalf and at the request of the client application; c) the server **application** independently **registering** its **notification** mechanism with an independent server discovery service; d) the server discovery service listening for the discovery packet, and

L18 ANSWER 39 OF 76 USPATFULL
PI US 5974541 19991026
TI GPIB system and method which provides asynchronous event notification
IN Hall, Mike, Round Rock, TX, United States
Bean, Clay, Austin, TX, United States
NCL NCLM: **712/228.000**
NCLS: **709/108.000; 710/100.000**
AB A GPIB system which includes asynchronous event notification. The GPIB **application** first provides a **notify request** to GPIB **driver** level **software**, preferably an ibnotify call or a GPIB Notify OLE control. The notify request includes a unit descriptor which uniquely identifies the GPIB device, event information regarding a plurality of GPIB events to monitor, a reference to a callback function in the GPIB application, and user defined reference data for the callback function. In response to the **notify request**, the GPIB **driver** level **software** begins monitoring events specified by the event information. When an event occurs which is being monitored, the GPIB driver level software recognizes the event and invokes the callback function. The invocation of the callback function is performed asynchronously to the GPIB application. The callback function may include one or more calls to one or more functions in the GPIB driver software. The callback function also uses the user defined reference data to aid in handling the event.
AB A GPIB system which includes asynchronous event notification. The GPIB **application** first provides a **notify request** to GPIB **driver** level **software**, preferably an ibnotify call or a GPIB Notify OLE control. The notify request includes a unit descriptor which uniquely identifies the GPIB device, event information regarding a plurality of GPIB events to monitor, a reference to a callback function in the GPIB application, and user defined reference data for the callback function. In response to the **notify request**, the GPIB **driver** level **software** begins monitoring events specified by the event information. When an event occurs which is being monitored, the GPIB driver level software recognizes the event and invokes the callback function. The invocation of the callback function is performed asynchronously to the GPIB application. The callback function may include one or more calls to one or more functions in the GPIB driver software. The callback function also uses the user defined reference data to aid in handling the event.
CLM What is claimed is:
1. A method of asynchronously notifying a GPIB application when one or more GPIB events occur in a GPIB system, wherein the system comprises the GPIB application which interfaces through GPIB driver software to a GPIB device, wherein the GPIB application and the GPIB driver software execute on a computer system, the method comprising: receiving an asynchronous **notify request** from the GPIB **application**, wherein the asynchronous **notify request** includes a unit descriptor, event **information** regarding a plurality of GPIB events to monitor, and a reference to a callback function in the GPIB application, wherein the unit descriptor uniquely identifies the GPIB device; and monitoring events specified by said event information, wherein said monitoring is performed in response to said receiving said asynchronous notify request, wherein said events comprise state changes of the GPIB device identified by the unit descriptor; determining that an event specified by said event information has occurred; invoking said callback function in response

to said determining that said event has occurred, wherein said invocation is performed asynchronously to the GPIB application.

4. The method of claim 1, further comprising: configuring the GPIB driver software to use interrupts prior to receiving the asynchronous **notify request** from the GPIB **application**;
said event occurring; generating an interrupt in response to said event occurring wherein said determining that an event specified by said event information has occurred includes: receiving the interrupt indicating that said event has occurred; and asynchronously notifying the GPIB application of said event.

12. The method of claim 1, further comprising: beginning an asynchronous I/O operation; receiving a **notify request** from the GPIB **application** after beginning the asynchronous I/O operation; and resynchronizing the GPIB driver level **software** in response to receiving the **notify request**.

14. A GPIB system which asynchronously notifies a GPIB application when one or more GPIB events o

L18 ANSWER 50 OF 76 USPATFULL
PI US 5835089 19981110
TI Application programming interface for shared address book services in a computer system
IN Skarbo, Rune A., Hillsboro, OR, United States
Elliott, John D., Aloha, OR, United States
NCL NCLM: 345/751.000
NCLS: **709/245.000; 709/328.000; 709/331.000**
AB An application programmers interface for shared address book services in a computer system that provides a register function call that enables a set of client communication application programs executing on the computer system to each **register** a **callback** function. The **application** programmers interface also provides a notify function call that enables an address book application program executing on the computer system to notify the appropriate client communication application program via the appropriate callback function that a user has selected a destination identifier for the corresponding communication type.
AB An application programmers interface for shared address book services in a computer system that provides a register function call that enables a set of client communication application programs executing on the computer system to each **register** a **callback** function. The **application** programmers interface also provides a notify function call that enables an address book application program executing on the computer system to notify the appropriate client communication application program via the appropriate callback function that a user has selected a destination identifier for the corresponding communication type.
CLM What is claimed is:
8. A method for sharing address book services for each of a set of client communication application programs in a computer system, the method comprising the steps of: registering a callback function with a shared address book service for each client communication application program executing on the computer system, each client communication application program servicing one or more of a set of communication types, and the shared address book service including an integrated address file having a plurality of addressees with addresses associated with different communication types; sensing a user selection through an address book application program wherein the user selection specifies one of said addresses with communication types; and invoking the **callback** function of the **requested client** communication **application program** with a communication type that corresponds to the communication type of the selected address, through the shared address book service, wherein the callback function passes the selected address to the requested client communication application program, initiating a communication transaction with the addressee of the selected address.

15. An apparatus for sharing address book services for each of a set of client communication application programs in a computer system, the apparatus comprising: means for **registering** a **callback** function for each **client** communication **application** program executing on the computer system with a shared address book service, each client communication application program servicing o

L18 ANSWER 51 OF 76 USPATFULL
PI US 5826253 19981020
TI Database system with methodology for notifying clients of any additions, deletions, or modifications occurring at the database server which affect validity of a range of data records cached in local memory buffers of clients
IN Bredenberg, David, Acton, MA, United States
NCL NCLM: **707/002.000**
NCLS: **707/001.000; 707/007.000; 707/008.000; 707/010.000; 707/100.000 ; 711/100.000; 711/103.000**
AB Client/server system and methods are described for providing a "cache range" to database clients. When one or more records in a cache range of a client change, a server in connection with the client sends a notification that the cache range has changed. Instead of the client taking a lock out for updating a record, the client simply indicates to the server which records it is interested in (e.g., via registering an event alerter request), whereupon the server manages its resources as necessary to notify the client of a change in one of the records which is of interest to the client. The server can simply take out "interest" locks on the corresponding records; these are not locks which exclude the resource from other clients. Other clients are not prevented from accessing this resource (i.e., the records which are of interest). The interest lock is employed in conjunction with the **registered** event alerter for **notifying** the **client** when a range of records of interest is updated by another client. In this manner, the server undertakes action to indicate to the client when the client local buffer might be stale, thereby providing improved local record buffer management in a client/server database context.
AB Client/server system and methods are described for providing a "cache range" to database clients. When one or more records in a cache range of a client change, a server in connection with the client sends a notification that the cache range has changed. Instead of the client taking a lock out for updating a record, the client simply indicates to the server which records it is interested in (e.g., via registering an event alerter request), whereupon the server manages its resources as necessary to notify the client of a change in one of the records which is of interest to the client. The server can simply take out "interest" locks on the corresponding records; these are not locks which exclude the resource from other clients. Other clients are not prevented from accessing this resource (i.e., the records which are of interest). The interest lock is employed in conjunction with the **registered** event alerter for **notifying** the **client** when a range of records of interest is updated by another client. In this manner, the server undertakes action to indicate to the client when the client local buffer might be stale, thereby providing improved local record buffer management in a client/server database context.
CLM What is claimed is:
1. In a multi-user computer system, the system including a database server maintaining on a storage device data records having data fields which store values that change in response to modifications posted by multiple clients in communication with said database server, each client maintaining a local memory buffer for caching a local copy of data records which have been retrieved, a method for notifying clients of any additions, deletions, or modifications occurring at the database server which affect validity of a range of data records

cached in the local memory buffer of clients, the method comprising: receiving at said database server a request from a particular client which defines a range of data records to fetch; in response to said request, (i) transferring a valid copy of data records within said range from the storage device of the server to the particular client, whereupon a valid copy of data records within said range exists in the local memory buffer of the particular client, and (ii) registering at the database server a **request** to **notify** the particular **client** of any event occurring at the database

L18 ANSWER 55 OF 76 USPATFULL
PI US 5805886 19980908
TI Method for notifying client applications of events in a shared application in a computer system
IN Skarbo, Rune A., Hillsboro, OR, United States
Elliott, John D., Aloha, OR, United States
NCL NCLM: **709/318.000**
AB A method for notifying a client application program of an event in a shared application program in a computer system is disclosed. A client application **program** executing in the computer system **registers** a **callback** function during startup. The shared **application** program senses the event and maps the event to one of a set of **registered client application programs**. The **callback** function of the appropriate **client application** program is then invoked such that the callback function passes parameters for the event to the client application program.
AB A method for notifying a client application program of an event in a shared application program in a computer system is disclosed. A client application **program** executing in the computer system **registers** a **callback** function during startup. The shared **application** program senses the event and maps the event to one of a set of **registered client application programs**. The **callback** function of the appropriate **client application** program is then invoked such that the callback function passes parameters for the event to the client application program.
CLM What is claimed is:
1. A method comprising: registering a callback function for each of one or more client application programs executing on a computer system; sensing an event in a shared application program; checking whether a client application program of the one or more client application programs is registered for a type indicator which maps to the event; invoking the callback function of the client application program if the client application program is registered for the type indicator, wherein the callback function passes a parameter for the event to the client application program if the client application program is registered for the type indicator; and launching a default client application program for the type indicator if no client application program is registered for the type indicator, and invoking a callback function of the default client application program after the default **client application program registers** a **callback** function.

6. An apparatus comprising: first logic to register a callback function for each of one or more client application programs executing on a computer system; second logic to sense an event in a shared application program, and to check whether a client application program of the one or more client application programs is registered for a type indicator which maps to the event, third logic to invoke the callback function of the client application program if the client application program is registered for the type indicator, wherein the callback function passes a parameter for the event to the client application program if the client application program is registered for the type indicator, and to launch a default client application program for the type indicator if no client application program is registered for the type indicator, and invoking a callback function of the default client application program after the default

client application program
registers a **callback** function.

11. A method for sending notification messages from a shared address book dynamic link library application to a collection of client applications, said method comprising: registering a callback function for one of said collection of client applications running on a computer system; sensing an event in said shared address book dynamic link library application; checking whether a client application of the collection of client applications is registered for a type indicator which maps to the event; invoking said callback function of said client application if said client application is registered for the tape indicator, wherein said callback function passes a parameter for said event to said client application; and launching a default client application for the type indicator if no client application is registered for the type indicator and invoking a callback function of

L18 ANSWER 66 OF 76 USPATFULL
PI US 5678002 19971014
TI System and method for providing automated customer support
IN Fawcett, Philip E., Duvall, WA, United States
Blomfield-Brown, Christopher, Seattle, WA, United States
NCL NCLM: 345/709.000
NCLS: **709/224.000; 714/046.000**
AB An automated system and method for diagnosing and resolving computer-related problems from a product support center. The traditional roles of the product engineer and customer are largely supplanted by execution of specialized client/server software programs on the respective computers. These programs exchange diagnostic and--in some cases--remedial data over the same telephone circuit used by the engineer and customer. The resulting system accelerates diagnosis and resolution of the customers' problems, reduces product support costs, and improves customer satisfaction.
CLM What is claimed is:
13. The method of claim 12 which includes calling said **registered** receive **callback** function each time the support **client** receives a message destined for the diagnostic interpreter, said calling serving to notify the diagnostic interpreter of the arrival of a message, and to initiate processing of the message.

NCL NCLM: 345/709.000
NCLS: **709/224.000; 714/046.000**|

L18 ANSWER 67 OF 76 USPATFULL
PI US 5555417 19960910
TI Method and apparatus for compiling computer programs with interprocedural register allocation
IN Odnert, Daryl, Boulder Creek, CA, United States
Santhanam, Vatsa, Sunnyvale, CA, United States
NCL NCLM: **717/159.000**
NCLS: **717/144.000; 717/146.000; 717/156.000; 717/158.000**
AB Optimization techniques are

L4 ANSWER 14 OF 57 USPATFULL
PI US 6249828 B1 20010619
TI Method for the hot swap of a mass storage adapter on a system including a statically loaded adapter driver
IN Wallach, Walter August, Los Altos, CA, United States
Khalili, Mehrdad, San Jose, CA, United States
Mahalingam, Mallikarjunan, Santa Clara, CA, United States
Reed, John M., Morgan Hill, CA, United States
NCL NCLM: **710/302.000**
NCLS: **713/001.000**; **713/100.000**
AB A software architecture for the hot add and swap of adapters. The software architecture allows users to replace failed components, upgrade
outdated components, and add new functionality, such as new network interfaces, disk interface adapters and storage, without impacting existing users. The software architecture supports the hot add and swap of off-the-shelf adapters, including those adapters that are programmable.
CLM What is claimed is:
5. The method of claim 1, **additionally** comprising creating an input/output request packet having a **callback** routine that is called to request the hot **swap** between the programmable mass storage adapter and the new programmable mass storage adapter.

12. The method of claim 8, **additionally** comprising creating an input/output request packet having a **callback** routine that is called to request the hot **swap** between the programmable mass storage adapter and the new programmable mass storage adapter.

18. The method of claim 15, **additionally** comprising creating an input/output request packet including a **callback** routine that is called to request the hot **swap** between the mass storage adapter and the new mass storage adapter.

NCL NCLM: **710/302.000**
NCLS: **713/001.000**; **713/100.000**

L4 ANSWER 42 OF 57 USPATFULL
PI US 5537597 19960716
TI Method and apparatus for supporting real mode card services clients with
a protected mode card services implementation
IN Sandage, David A., Forrest Grove, OR, United States
NCL NCLM: **717/162.000**
NCLS: **709/321.000; 713/002.000**
AB PCMCIA defines a standard interface for small portable computer peripherals. Part of the PCMCIA specification defines a Card Services software layer. The current PCMCIA specification defines a Card Services
layer that provides Card Services to clients that wish to use PCMCIA cards. A Card Services Compatibility Driver is defined that ensures compatibility with real mode clients. The Card Services Compatibility Driver is a device driver or TSR that loads near the beginning of the computer system boot procedure. The Card Services Compatibility Driver simulates a full implementation of Card Services by supporting only the functions that are legal when no cards are installed in the system even though cards may be installed in the system. Later, a full protected mode implementation of Card Services is loaded into the system. The Card
Services Compatibility Driver transfers all the state information it collects during the computer system boot procedure to the full protected
mode implementation of Card Services. After receiving this information, the full protected mode implementation of Card Services takes control and services all later Card Services requests. The full protected mode implementation of Card Services also informs the registered Card Services clients of any cards **inserted** in the system by making card **insertion callbacks**.
AB PCMCIA defines a standard interface for small portable computer peripherals. Part of the PCMCIA specification defines a Card Services software layer. The current PCMCIA specification defines a Card Services
layer that provides Card Services to clients that wish to use PCMCIA cards. A Card Services Compatibility Driver is defined that ensures compatibility with real mode clients. The Card Services Compatibility Driver is a device driver or TSR that loads near the beginning of the computer system boot procedure. The Card Services Compatibility Driver simulates a full implementation of Card Services by supporting only the functions that are legal when no cards are installed in the system even though cards may be installed in the system. Later, a full protected mode implementation of Card Services is loaded into the system. The Card
Services Compatibility Driver transfers all the state information it collects during the computer system boot procedure to the full protected
mode implementation of Card Services. After receiving this information, the full protected mode implementation of Card Services takes control and services all later Card Services requests. The full protected mode implementation of Card Services also informs the registered Card Services clients of any cards **inserted** in the system by making card **insertion callbacks**.
CLM What is claimed is:
3. The method of providing Card Services functions as claimed in claim 2
wherein said step of informing said Card Services client of any PCMCIA cards installed in said computer system comprises calling a card

insertion callback routine in said Card Services client informing said Card Services client about a PCMCIA card insertion.

13. The computer system as claimed in claim 12 wherein said protected mode Card Services implementation informs said Card Services client of any PCMCIA cards installed in said PCMCIA adapter by calling a card **insertion callback** routine in said Card Services client.

NCL NCLM: **717/162.000**
NCLS: **709/321.000; 713/002.000**

L5 ANSWER 2 OF 49 USPATFULL
PI US 2002087746 A1 20020704
TI Media manager for controlling autonomous media devices within a network environment and managing the flow and format of data between the devices
IN Ludtke, Harold Aaron, San Jose, CA, UNITED STATES
Fairman, Bruce, Woodside, CA, UNITED STATES
Smyers, Scott, San Jose, CA, UNITED STATES
NCL NCLM: **710/001.000**
AB A media manager provides data flow management and other services for client applications on devices coupled together within a network. Preferably, these devices are coupled together within an IEEE 1394-1995 serial bus network. A device control module is generated for each available device for providing an abstraction for all of the capabilities and requirements of the device including the appropriate control protocol, physical connections and connection capabilities for the device. The media manager also manages the flow and format of data transfers between the devices on the network. Through an interface, a user accesses the media manager and enters functions which are to be completed using the devices coupled together on the network. If the appropriate devices are available, the media manager controls and manages the completion of the requested task. If the appropriate devices are not available, but the required subdevices are available in multiple devices, the media manager forms a virtual device from subdevices in multiple devices in order to complete the requested task. Once the appropriate devices and subdevices are assigned to a task, the media manager determines if the data to be transmitted needs to be converted from one format into another format. If necessary, the media manager will also control the format conversion during the data transfer operation. The media manager also provides network enumeration and registry searching capabilities for client applications to find available services, physical devices and virtual devices.
DETD [0073] After completing the step of registering with the media manager, the application 48 then will obtain the available DCMs 56 at the step 146. By obtaining the available DCMs 56, the application 48 will know the other types of devices which are **coupled** within the network. This step is composed of a series of sub-steps. An iterative **callback** model is used as the method of data transfer for transferring the data to the application 48. The client application 48 first gives the media manager an address of a **callback** function. The application 48 then enters a loop and repeatedly requests information about the next module from the media manager until there are no remaining DCMs. The media manager prepares a data structure with the necessary information and transfers it to the application 48 via the **callback** function. Once the information about each specific DCM 56 is received, the application 48 then copies the information which it requires. This process is repeated until all of the available DCMs 56 have been received by the application 48. Within an alternate embodiment of the present invention, the client application queries the system registry, requesting a handle to each of the available DCMs 56.
DETD [0077] The parameter deviceIndex of the **callback** function is the index value of the loop which the application 48 enters to obtain information about the available DCMs 56. The loop is bounded by the number of available DCMs 56. This parameter is passed back to the application 48 in the **callback** function so that the application 48 can save this parameter along with the other information passed into the **callback** function. This index

value is useful in other calls to the media manager by the application 48, when inquiring about a specific DCM 56. In **addition**, this index value will be used when notifying the application 48 that a device has disappeared after it was unplugged or **disconnected** from the network. The application 48 will store this index value for each DCM 56 within a dedicated field in its private data structure.

NCL NCLM: **710/001.000**

=> d pn,ti,in,ncl,ab,hit 2, 15, 23, 27, 41, 42

L5 ANSWER 2 OF 49 USPATFULL
PI US 2002087746 A1 20020704
TI Media manager for controlling autonomous media devices within a network environment and managing the flow and format of data between the devices
IN Ludtke, Harold Aaron, San Jose, CA, UNITED STATES
Fairman, Bruce, Woodside, CA, UNITED STATES
Smyers, Scott, San Jose, CA, UNITED STATES
NCL NCLM: **710/001.000**
AB A media manager provides data flow management and other services for client applications on devices coupled together within a network. Preferably, these devices are coupled together within an IEEE 1394-1995 serial bus network. A device control module is generated for each available device for providing an abstraction for all of the capabilities and requirements of the device including the appropriate control protocol, physical connections and connection capabilities for the device. The media manager also manages the flow and format of data transfers between the devices on the network. Through an interface, a user accesses the media manager and enters functions which are to be completed using the devices coupled together on the network. If the appropriate devices are available, the media manager controls and manages the completion of the requested task. If the appropriate devices are not available, but the required subdevices are available in multiple devices, the media manager forms a virtual device from subdevices in multiple devices in order to complete the requested task. Once the appropriate devices and subdevices are assigned to a task, the media manager determines if the data to be transmitted needs to be converted from one format into another format. If necessary, the media manager will also control the format conversion during the data transfer operation. The media manager also provides network enumeration and registry searching capabilities for client applications to find available services, physical devices and virtual devices.
DETD [0073] After completing the step of registering with the media manager, the application 48 then will obtain the available DCMs 56 at the step 146. By obtaining the available DCMs 56, the application 48 will know the other types of devices which are **coupled** within the network. This step is composed of a series of sub-steps. An iterative **callback** model is used as the method of data transfer for transferring the data to the application 48. The client application 48 first gives the media manager an address of a **callback** function. The application 48 then enters a loop and repeatedly requests information about the next module from the media manager until there are no remaining DCMs. The media manager prepares a data structure with the necessary information and transfers it to the application 48 via the **callback** function. Once the information about each specific DCM 56 is received, the application 48 then copies the information which it requires. This process is repeated until all of the available DCMs 56 have been received by the application 48. Within an alternate embodiment of the present invention, the client application queries the system registry, requesting a handle to each of the available DCMs 56.
DETD [0077] The parameter deviceIndex of the **callback** function is the index value of the loop which the application 48 enters to obtain information about the available DCMs 56. The loop is bounded by the number of available DCMs 56. This parameter is passed back to the

application 48 in the **callback** function so that the application 48 can save this parameter along with the other information passed into the **callback** function. This index value is useful in other calls to the media manager by the application 48, when inquiring about a specific DCM 56. In **addition**, this index value will be used when notifying the application 48 that a device has disappeared after it was unplugged or **disconnected** from the network. The application 48 will store this index value for each DCM 56 within a dedicated field in its private data structure.

NCL NCLM: **710/001.000**

L5 ANSWER 15 OF 49 USPATFULL

PI US 6311242 B1 20011030

TI Method and apparatus for supporting dynamic insertion and removal of PCI devices

IN Falkenburg, David R., San Jose, CA, United States
Wynne, Edwin, Plymouth, MN, United States
Thaler, Andrew, Saginaw, MI, United States

NCLS: 345/503.000; 345/520.000; **361/118.000**; **361/686.000**; **361/802.000**; 707/500.100; **710/002.000**; **710/008.000**; **710/017.000** ; **710/027.000**; **710/072.000**; **710/100.000**; **710/104.000**; **710/124.000** ; **710/300.000**; **710/301.000**; **710/302.000**; **710/305.000**; **710/313.000**

AB Improved techniques for controlling buses of a computer system are disclosed such that peripheral devices (and/or their associated buses) can be connected or disconnected to the computer system while the computer system is active. The peripheral devices are connected to the computer system by being inserted into a slot or other receptacle of the computer system. The peripheral devices are disconnected from the computer system by being removed from a slot or other receptacle of the computer system. The slots or receptacles typically includes connectors designed to receive peripheral devices, such as PC CARD slots, expansion bays, and the like. Given that the peripheral devices can be inserted or removed while the computer system is active is active, the computer system according to the invention permits "hot-plugging" of peripheral devices. The invention is particularly well suited for controlling PCI buses for peripheral devices connecting to a computer system by way of peripheral ports.

DETD On the other hand, when the decision block 404 determines that the bus ID is valid, then the enumerate processing 400 probes 408 the **added** PCI bus to identify PCI devices on the **added** PCI bus. Then, for each of the identified PCI devices, an associated **callback** function can be performed 410. The performance of the associated **callback** functions are used to provide **additional** processing for the PCI devices. For example, the **additional** processing can include **additional** customized probing routines that can be supplied with the PCI devices. These **additional** customized probing routines can operate in various ways and typically serve to further examine or identify the particular identified PCI devices. By allowing for the separate **callback** functions, the probing process in able to be easily updated and changed without affecting the architecture of the enumerate processing 400.

NCLS: 345/503.000; 345/520.000; **361/118.000**; **361/686.000**; **361/802.000**; 707/500.100; **710/002.000**; **710/008.000**; **710/017.000** ; **710/027.000**; **710/072.000**;

710/100.000; 710/104.000; 710/124.000
; 710/300.000; 710/301.000;
710/302.000; 710/305.000; 710/313.000

L5 ANSWER 23 OF 49 USPATFULL
PI US 6160796 20001212
TI Method and system for updating device identification and status information after a local bus reset within a home audio/video network
IN Zou, Feng (Frank), Milpitas, CA, United States
NCL NCLM: 370/257.000
NCLS: **710/104.000**
AB A method and system for updating device identification and status information in response to a local bus reset within a home audio/video network. Several consumer electronics products, e.g., television, VCR, tuner, set-top box (e.g., intelligent receiver/decoder, IRD), DVTRs, PCs, DVD players (digital video disk), etc., can be **coupled** within the network to communicate together via a standard bus (e.g., IEEE 1394 serial communication bus). In one embodiment, the communication architecture used is the home audio/visual initiative (HAVI) format. The HAVI network offers unique advantages consumer electronic vendors because the architecture offers for the home network many of the advantages of existing computer system networks. Namely, interconnected devices can share resources and provide open, well defined APIs that allow ease of development for third party developers. The present invention provides a mechanism whereby devices of the network are informed of the current status of the network after a local bus reset caused when a device is **inserted** into the network or when a device is **removed** from the network. After a reset, a new GUID list is generated by a driver and passed to a high level program. The high level program then compares the new GUID list with its own copy of an older version and generates a list of devices **added** to the network after the bus reset and a list of devices **removed** from the network after the bus reset. This information is then forwarded to all devices on the network that previously specified certain **call back** information regarding current device status.
AB A method and system for updating device identification and status information in response to a local bus reset within a home audio/video network. Several consumer electronics products, e.g., television, VCR, tuner, set-top box (e.g., intelligent receiver/decoder, IRD), DVTRs, PCs, DVD players (digital video disk), etc., can be **coupled** within the network to communicate together via a standard bus (e.g., IEEE 1394 serial communication bus). In one embodiment, the communication architecture used is the home audio/visual initiative (HAVI) format. The HAVI network offers unique advantages consumer electronic vendors because the architecture offers for the home network many of the advantages of existing computer system networks. Namely, interconnected devices can share resources and provide open, well defined APIs that allow ease of development for third party developers. The present invention provides a mechanism whereby devices of the network are informed of the current status of the network after a local bus reset caused when a device is **inserted** into the network or when a device is **removed** from the network. After a reset, a new GUID list is generated by a driver and passed to a high level program. The high level program then compares the new GUID list with its own copy of an older version and generates a list of devices **added** to the network after the bus reset and a list of devices **removed** from the network after the bus reset. This information is then forwarded to all devices on the network that previously

specified certain **call back** information regarding current device status.

SUMM Devices can be **removed** from or **inserted** into the network while the network is otherwise operable (e.g., hot modifications are allowed). Therefore, it is important that devices be informed of the current status of other devices on the network following a bus reset. The present invention provides a mechanism whereby devices of the network are informed of the current status of the network after a local bus reset which is caused when a device is **inserted** into the network (including power on) or when a device is **removed** from the network (including power down). In accordance with the present invention, after a reset, a new GUID list is generated by a driver program and passed to a high level program. The high level program then compares the new (or "current") GUID list with its own copy of the last GUID list version and generates a list of devices **added** to the network after the bus reset and a list of devices **removed** from the network after the bus reset. This information is then forwarded to all devices on the network that previously specified certain **call back** information regarding current device status.

SUMM In one embodiment, the high level program is a communications media manager (CMM) that maintains a copy of the available devices (nodes) in the network in the form of a GUID list. The GUID is a persistent 64-bit device identification number that includes a vendor code and a chip series code and thereby uniquely identifies every device in the network. The device or program that has interest in knowing what devices are new and what devices have been **removed** from the network provides a **call back** handler to the CMM. When an IEEE 1394 compatible device (or several devices) is **inserted** in or **removed** from the local bus, or any other topology change occurs, a bus reset is generated within the network. A low level driver program, a link driver, catches the bus reset, updates the available devices and passes this information (the new GUID list) to the CMM. The CMM compares its version of the GUID list with the new GUID list and creates new information including the new and **removed** devices. This status information is passed through the **call back** handler to each interested object. The present invention thereby provides an efficient mechanism for new and/or **removed** device notification and **removes** the burden from high level software of maintaining a list of recently **added** or **removed** devices.

DETD One of the advanced features the 1394 bus provides to the HAVI architecture is its support for dynamic device actions such as hot **plugging** (device **insertion** or power up) and unplugging (device **removal** or power down). To fully support this to the user level, high level software clients need to be aware of these environment changes and the present invention provides this information. The CMM 250 works with the Event Manager 214 to detect and announce these dynamic bus changes using device status tables (FIG. 12A, FIG. 12B, FIG. 12C). Since any topology change within 1394 bus will cause a bus reset to occur, the CMM 250 is informed of these changes and notifies the Event Manager 214 of these changes along with the information about devices that have disappeared as well as those that have become available. The Event Manager 214 then distributes the related event to all interested HAVI entities or applications that previously established the proper **call back** handlers.

DETD This command is used by the Event Manager 214 to get a list of new and **removed** devices when bus reset occurs. The Event Manager 214 enrolls such request to the CMM 250 to get the service. The parameter oid is the caller's object identifier and **callback**.sub.-- handler is the **callback** function provided by the caller. The caller (typically Event Manager 214) enrolls its OID and a **callback** handler to the CMM 250 so that when bus reset occurs, the CMM 250 invokes the **callback** function with a list of devices (device status table) that are either new or gone.

DETD The Event Manager 214 of FIG. 4 is designed to support a one-to-many model of communication. It is built above the basic messaging system 212. Objects register interest in events which are then delivered to their event **call back** entry. For instance, objects interested in device status information register special **call back** handlers for this information with the Event Manager 214. Objects can **attach** event filters to the **callback** entries that allow them to filter out events that they are not currently interested in. For example an object may register for all events from display devices, but filter out those display devices it is not currently using.

DETD FIG. 12B illustrates a subset status table 650 generated by the CMM 250 which contains entries 652a-652b indicating only which devices have been **added** to the local bus 30 since the last GUID list 510 was compiled. Likewise, FIG. 12C illustrates a subset status table 660 generated by the CMM 250 which contains entries (662a) indicating only which devices have been **removed** from the local bus 30 since the last GUID list 510 was compiled. In one embodiment, the GUID tables 650-660 are compiled by the CMM 250 and stored in computer readable memory. Device status tables 630, 650 and 660 are forwarded to objects within the HAVI architecture that establish **callback** handlers within the CMM 250 to receive this information.

CLM What is claimed is:

1. A method of providing device status information within a communication network, said method comprising the steps of: a) constructing a current list of global unique identifiers (GUIDs) by ordering respective GUIDs within said current list of GUIDs according to their associated physical identifier values where n said physical identifier values are assigned by a local bus of said network and each GUID of said current list of GUIDs identifies a unique device **coupled** within said network; b) forwarding said current list of GUIDs to a high level software program; c) said high level software program comparing said current list of GUIDs with a previous list of GUIDs to generate a list of newly **added** devices to said network and of newly **removed** devices from said network; d) forwarding said list to a device within said network that previously established a **call back** handler with said high level software program for device status information; and e) repeating steps a)-d) in response to each local bus reset.

7. A method of providing device status information within a communication network, said method comprising the steps of: a) constructing a current list of global unique identifiers (GUIDs) by ordering respective GUIDs within said current list of GUIDs according to their associated physical identifier values, said physical identifier values assigned by a local bus of said network and wherein each GUID of said current list of GUIDs identifies a unique device **coupled** within said network, said step a) performed by first software; b) said first software forwarding said current list of GUIDs

to second software; c) said second software comparing said current list of GUIDs with a previous list of GUIDs to generate a first list of newly **added** devices to said communication network and a second list of newly **removed** devices from said network; d) forwarding said first list and said second list to a device within said communication network that previously established a **call back** handler with said second software for device status information; and e) repeating steps a)-d) in response to each local bus reset.

13. An electronic system having a processor, a bus and a computer readable memory unit, said system **coupled** to a local bus, said memory unit having instructions stored therein that implement a method of providing device status information, said method comprising the steps of: a) constructing a current list of global unique identifiers (GUIDs) by ordering respective GUIDs within said current list of GUIDs according to their associated physical identifier values wherein said physical identifier values are assigned by said local bus and each GUID of said current list of GUIDs identifies a unique device; b) forwarding said current list of GUIDs to a high level software program; c) said high level software program comparing said current list of GUIDs with a previous list of GUIDs to generate a list of newly **added** devices to said network and of newly **removed** devices from said network; d) forwarding said list to a device within said network that previously established a **call back** handler with said high level software program for device status information; and e) repeating steps a)-d) in response to each local bus reset.

19. An apparatus for providing device status information within a communication network, said apparatus comprising: a) means for constructing a current list of global unique identifiers (GUIDs) by ordering respective GUIDs within said current list of GUIDs according to their associated physical identifier values wherein said physical identifier values are assigned by a local bus of said network and each GUID of said current list of GUIDs identifies a unique device **coupled** within said network; b) means for forwarding said current list of GUIDs to a high level software program; c) said high level software program comparing said current list of GUIDs with a previous list of GUIDs to generate a list of newly **added** devices to said network and of newly **removed** devices from said network; d) said high level software program forwarding said list to a device within said network that previously established a **call back** handler with said high level software program for device status information; and e) means for repeating steps a)-d) in response to each local bus reset.

NCL NCLM: 370/257.000
NCLS: **710/104.000**|

L5 ANSWER 27 OF 49 USPATFULL
PI US 6038625 20000314
TI Method and system for providing a device identification mechanism within a consumer audio/video network
IN Ogino, Hiroshi, Sunnyvale, CA, United States
Zou, Feng (Frank), Milpitas, CA, United States
NCL NCLM: **710/104.000**
NCLS: 709/328.000; **710/008.000**; **710/009.000**; **710/010.000**; 712/001.000; 712/208.000

AB A method and system for providing a device identification mechanism within a consumer electronics based audio/video network. Several consumer electronics products, e.g., television, VCR, tuner, set-top box (e.g., intelligent receiver/decoder, IRD), DVTRs, PCs, DVD players (digital video disk), etc., can be coupled within the network to communicate together via a standard bus (e.g., IEEE 1394 serial communication bus). In one embodiment, the HAVI network offers unique advantages consumer electronic vendors because the architecture offers for the home network many of the advantages of existing computer system networks. Specifically, interconnected devices can share resources and provide open, well defined APIs that allow ease of development for third party developers. The present invention provides a mechanism whereby a global unique identifier (GUID) is associated with each device of the HAVI network. A low level driver constructs a GUID list of each device on the HAVI network. The order of the GUID entries in the GUID list (e.g., the index) matches the physical identifiers assigned to the devices by the 1394 serial bus. Although the physical identifiers can change on bus reset, the GUID values are constant and are used for device communication. Speed map and topology map information is maintained based on the physical identifier information and therefore translations between GUIDs and physical identifiers are efficiently performed by the present invention when referencing speed map and topology information for an application.

DETD One of the advanced features the 1394 bus provides to the HAVI architecture is its support for dynamic device actions such as hot **plugging** (device **insertion** or power up) and unplugging (device **removal** or power down). To fully support this to the user level, high level software clients need to be aware of these environment changes and the present invention provides this information. The CMM 250 works with the Event Manager 214 to detect and announce these dynamic bus changes using device status tables (FIG. 12A, FIG. 12B, FIG. 12C). Since any topology change within 1394 bus will cause a bus reset to occur, the CMM 250 is informed of these changes and notifies the Event Manager 214 of these changes along with the information about devices that have disappeared as well as those that have become available. The Event Manager 214 then distributes the related event to all interested HAVI entities or applications that previously established the proper **call back** handlers.

DETD This command is used by the Event Manager 214 to get a list of new and **removed** devices when bus reset occurs. The Event Manager 214 enrolls such request to the CMM 250 to get the service. The parameter oid is the caller's object identifier and **callback**.sub.-- handler is the **callback** function provided by the caller. The caller (typically Event Manager 214) enrolls its OID and a **callback** handler to the CMM 250 so that when bus reset occurs, the CMM 250 invokes the **callback** function with a list of devices (device status table) that are either new or gone.

DETD The Event Manager 214 of FIG. 4 is designed to support a one-to-many model of communication. It is built above the basic messaging system 212. Objects register interest in events which are then delivered to their event **call back** entry. For instance, objects interested in device status information register special **call back** handlers for this information with the Event Manager 214. Objects can **attach** event filters to the **callback** entries that allow them to filter out events that they are not currently interested in. For example an object may register for all events from display devices, but filter out those display devices it is not currently using.

DETD FIG. 12B illustrates a subset status table 650 generated by the CMM 250 which contains entries 652a-652b indicating only which devices have been **added** to the local bus 30 since the last GUID list 510 was compiled. Likewise, FIG. 12C illustrates a subset status table 660 generated by the CMM 250 which contains entries (662a) indicating only which devices have been **removed** from the local bus 30 since the last GUID list 510 was compiled. In one embodiment, the GUID tables 650-660 are compiled by the CMM 250 and stored in computer readable memory. Device status tables 630, 650 and 660 are forwarded to objects within the HAVI architecture that establish **callback** handlers within the CMM 250 to receive this information.

NCL NCLM: **710/104.000**
NCLS: 709/328.000; **710/008.000**; **710/009.000**; **710/010.000**; 712/001.000; 712/208.000

L5 ANSWER 41 OF 49 USPATFULL
PI US 5537597 19960716
TI Method and apparatus for supporting real mode card services clients with a protected mode card services implementation
IN Sandage, David A., Forrest Grove, OR, United States
NCL NCLM: 717/162.000
NCLS: 709/321.000; **713/002.000**
AB PCMCIA defines a standard interface for small portable computer peripherals. Part of the PCMCIA specification defines a Card Services software layer. The current PCMCIA specification defines a Card Services layer that provides Card Services to clients that wish to use PCMCIA cards. A Card Services Compatibility Driver is defined that ensures compatibility with real mode clients. The Card Services Compatibility Driver is a device driver or TSR that loads near the beginning of the computer system boot procedure. The Card Services Compatibility Driver simulates a full implementation of Card Services by supporting only the functions that are legal when no cards are installed in the system even though cards may be installed in the system. Later, a full protected mode implementation of Card Services is loaded into the system. The Card Services Compatibility Driver transfers all the state information it collects during the computer system boot procedure to the full protected mode implementation of Card Services. After receiving this information, the full protected mode implementation of Card Services takes control and services all later Card Services requests. The full protected mode implementation of Card Services also informs the registered Card Services clients of any cards **inserted** in the system by making card **insertion callbacks**.
AB PCMCIA defines a standard interface for small portable computer peripherals. Part of the PCMCIA specification defines a Card Services software layer. The current PCMCIA specification defines a Card Services layer that provides Card Services to clients that wish to use PCMCIA cards. A Card Services Compatibility Driver is defined that ensures compatibility with real mode clients. The Card Services Compatibility Driver is a device driver or TSR that loads near the beginning of the computer system boot procedure. The Card Services Compatibility Driver simulates a full implementation of Card Services by supporting only the functions that are legal when no cards are installed in the system even though cards may be installed in the system. Later, a full protected mode implementation of Card Services is loaded into the system. The Card Services Compatibility Driver transfers all the state information it collects during the computer system boot procedure to the full protected mode implementation of

Card Services. After receiving this information, the full protected mode implementation of Card Services takes control and services all later Card Services requests. The full protected mode implementation of Card Services also informs the registered Card Services clients of any cards **inserted** in the system by making card **insertion callbacks**.

SUMM For example, when a hardware event from a PCMCIA card socket occurs, the client's **callback** routine performs processing associated with the event. Events include hardware events such as card **insertion/removal** or low battery and software events such as a client requesting exclusive use of a card. Upon card **insertion**, a registered client might use the **callback** routine to query the card to determine the card's type. If card was the correct type it would then proceed to configure the card. The card's onboard memory, I/O, and interrupt must be set up to fit the system into which the card has been **plugged**. Card Services functions enable clients to configure cards by programming the card hardware registers and the PCMCIA adapter.

SUMM These and other objectives are accomplished by a Card Services Compatibility Driver. The Card Services Compatibility Driver is a device driver or TSR that loads near the beginning of the computer system boot procedure. The Card Services Compatibility Driver simulates a full implementation of Card Services by supporting only the functions that are legal when no cards are installed in the system even though cards may be installed in the system. Later, a full protected mode implementation of Card Services is loaded into the system. The Card Services Compatibility Driver transfers all the state information it has collected during the computer system boot procedure to the full protected mode implementation of Card Services. After receiving this information, the full protected mode implementation of Card Services takes control and services all future Card Services requests. The full protected mode implementation of Card Services also informs the registered Card Services clients of any cards **inserted** in the system by making card **insertion callbacks**.

DETD FIG. 3a illustrates how the program entities interact during the computer system boot before the Windows.TM. operating environment has been loaded. As illustrated in FIG. 3a, all requests for Card Services functions are handled by a Card Services Compatibility Driver 50. The real mode application programs, device drivers, and TSRs can register as Card Services client. The programs can also get status and perform some of the physical to logical mappings. However, the Card Services clients will not receive any card **insertion callbacks** at this time.

DETD FIG. 3b illustrates how the program entities interact during the computer system boot after the Windows.TM. operating environment and the full protected mode Card Services implementation 51 have been loaded. After the computer loads the full protected mode Card Services implementation, the full protected mode Card Services implementation 51 requests the Card Services Compatibility Driver 50 to provide all the information that the Card Services Compatibility Driver 50 collected during the computer system boot. This information includes: the socket services data structure; a list of registered Card Services clients (including handles, **callback** entry points, client data, global event mask); a list of registered erase queues; a list of registered timers (with amount of time remaining on each timer); and the available resources (if appropriate for the operating

environment). Note that the Card Services Compatibility Driver 50 and the full protected mode Card Services implementation 51 must perform physical/logical socket and window mapping in the same manner in order for the transition to be transparent. After obtaining the information, the full protected mode Card Services implementation 51 will take over the INT 1A entry point from the Card Services Compatibility Driver 50 and assume the role of Card Services provider to all Card Services clients. The protected mode Card Services implementation 51 then informs the registered Card Services clients about any cards present in the system by performing card **insertion callbacks** for all the cards present in the system.

DETD However, if one or more cards are present in the system then the protected-mode implementation of Card Services may need to inform some of the registered clients of Card Services. The protected-mode implementation of Card Services first examines the list of registered clients to find the clients that wish to know about card **insertions**. The protected-mode implementation of Card Services then performs "**callbacks**" to those clients informing the clients that a card has been **inserted**, as stated in step 117. After informing all registered clients that wish to know about card **insertions** about all the cards present in the system, then the Card Services initialization procedure is complete.

CLM What is claimed is:

3. The method of providing Card Services functions as claimed in claim 2 wherein said step of informing said Card Services client of any PCMCIA cards installed in said computer system comprises calling a card **insertion callback** routine in said Card Services client informing said Card Services client about a PCMCIA card **insertion**.

13. The computer system as claimed in claim 12 wherein said protected mode Card Services implementation informs said Card Services client of any PCMCIA cards installed in said PCMCIA adapter by calling a card **insertion callback** routine in said Card Services client.

NCL NCLM: 717/162.000
NCLS: 709/321.000; **713/002.000**|

L5 ANSWER 42 OF 49 USPATFULL
PI US 5519851 19960521
TI Portable PCMCIA interface for a host computer
IN Bender, Michael S., Boulder Creek, CA, United States
McCallum, Douglas, Louisville, CO, United States
Patton, Jr., Charles F., Dublin, CA, United States
Vo, Duong M., Milpitas, CA, United States
NCL NCLM: **710/301.000**
AB A portable PCMCIA interface for a host computer having a system bus. In one embodiment, the host computer is a SPARC based computer having an SBus and running the UNIX operating system. The PCMCIA interface provides a user application with access to a PCMCIA card. In this embodiment, the PCMCIA interface includes software and hardware components. The software component includes a hardware-independent portion and a hardware-dependent portion. By implementing the software in a suitable high level language such as "C", the software can be easily ported to other hardware platforms by merely adapting the hardware-dependent portion. The hardware component includes an ASIC coupled between the system bus and a couple of PCMCIA sockets. In some embodiments, the hardware also includes a 5 volt to 12 volt DC--DC

converter between the system bus and the PCMCIA sockets.

DETD Next, in order to support a PCMCIA specified event **callback** feature, event manager 230 is loaded as a separate STREAMS, enabling nexus driver 210 to communicate PCMCIA events, such as PC card **insertion/removal** (hot-**plugging**), to user application 31. (A STREAMS is a UNIX full-duplex **connection** between a process and a device driver). Hence, nexus driver 210 is provided an efficient event **callback** mechanism to user application 31, with event manager 230 observing the events and managing both adapter 100 and PC card 110 via adapter driver 250. Event manager 230 is an efficient solution because the number of PC cards available make it unwieldy and inefficient for user application 31 to poll for every possible PCMCIA card. More importantly, event manager 230 typically provides the sole mechanism (generally unsupported by UNIX) for implementing the PCMCIA specified channel for unsolicited feedback to user application 31.

DETD For example, when PC card 110 has been successfully **inserted** into PCMCIA socket 107, nexus driver 210 receives a "card **insertion**" event notification from adapter driver 250, enabling nexus driver 210 to keep track of which type of PC card is present in socket 107. Subsequently, a corresponding PC card driver, e.g., driver 260, is loaded and nexus driver 210 updates the corresponding driver apparatus structure, e.g., structure 251, thereby forming an association between PCMCIA socket 107 and PC card 110. A discussion of the PCMCIA interface hardware for supporting this event **callback** feature is included in a detailed hardware description of adapter 100 below.

NCL NCLM: **710/301.000**

L7 ANSWER 12 OF 55 USPATFULL
PI US 6311242 B1 20011030
TI Method and apparatus for supporting dynamic insertion and removal of PCI
devices
IN Falkenburg, David R., San Jose, CA, United States
Wynne, Edwin, Plymouth, MN, United States
Thaler, Andrew, Saginaw, MI, United States
NCLS: 345/503.000; 345/520.000; 361/118.000; 361/686.000; 361/802.000;
707/500.100; 710/002.000; 710/008.000
; 710/017.000; 710/027.000;
710/072.000; 710/100.000; 710/104.000
; 710/124.000; 710/300.000;
710/301.000; 710/302.000; 710/305.000
; 710/313.000
AB Improved techniques for controlling buses of a computer system are disclosed such that peripheral devices (and/or their associated buses) can be connected or disconnected to the computer system while the computer system is active. The peripheral devices are connected to the computer system by being inserted into a slot or other receptacle of the
computer system. The peripheral devices are disconnected from the computer system by being removed from a slot or other receptacle of the computer system. The slots or receptacles typically includes connectors designed to receive peripheral devices, such as PC CARD slots, expansion
bays, and the like. Given that the peripheral devices can be inserted or
removed while the computer system is active is active, the computer system according to the invention permits "hot-plugging" of peripheral devices. The invention is particularly well suited for controlling PCI buses for peripheral devices connecting to a computer system by way of peripheral ports.
DETD The removal processing uses a two step **notification**
process to initially **notify** the appropriate
device driver of the **removal** of the PCI device so that in a "hot-removal" or "hot-unplugged" situation, the appropriate device driver is almost instantly notified that the PCI device has been removed. Then, the appropriate device driver will not attempt to utilize
the PCI device which is no longer present in the slot or receptacle of the computer system. Thereafter, once the computer system has processing
availability, the computer system can perform the administrative processing tasks associated with the removal of the previously assigned structures within the computer system for use by the previously inserted, now removed PCI device. Hence, the major processing requirements of the removal processing 300 are deferred until the computer system has sufficient processing availability so that the performance of the computer system, particularly if real-time performance, is not affected. Further, the removal of the previously assigned structures (e.g., device driver, memory space and name registry) frees up system resources for use by entities without having to reboot the system or switch between active and inactive modes.
NCLS: 345/503.000; 345/520.000; 361/118.000; 361/686.000; 361/802.000;
707/500.100; 710/002.000; 710/008.000
; 710/017.000; 710/027.000;
710/072.000; 710/100.000; 710/104.000

; 710/124.000; 710/300.000;
710/301.000; 710/302.000; 710/305.000
; 710/313.000

L7 ANSWER 12 OF 55 USPATFULL
PI US 6311242 B1 20011030
TI Method and apparatus for supporting dynamic insertion and removal of PCI devices
IN Falkenburg, David R., San Jose, CA, United States
Wynne, Edwin, Plymouth, MN, United States
Thaler, Andrew, Saginaw, MI, United States
NCLS: 345/503.000; 345/520.000; 361/118.000; 361/686.000; 361/802.000; **707/500.100; 710/002.000; 710/008.000 ; 710/017.000; 710/027.000; 710/072.000; 710/100.000; 710/104.000 ; 710/124.000; 710/300.000; 710/301.000; 710/302.000; 710/305.000 ; 710/313.000**
AB Improved techniques for controlling buses of a computer system are disclosed such that peripheral devices (and/or their associated buses) can be connected or disconnected to the computer system while the computer system is active. The peripheral devices are connected to the computer system by being inserted into a slot or other receptacle of the computer system. The peripheral devices are disconnected from the computer system by being removed from a slot or other receptacle of the computer system. The slots or receptacles typically includes connectors designed to receive peripheral devices, such as PC CARD slots, expansion bays, and the like. Given that the peripheral devices can be inserted or removed while the computer system is active is active, the computer system according to the invention permits "hot-plugging" of peripheral devices. The invention is particularly well suited for controlling PCI buses for peripheral devices connecting to a computer system by way of peripheral ports.
DETD The removal processing uses a two step **notification process** to initially **notify** the appropriate **device** driver of the **removal** of the PCI device so that in a "hot-removal" or "hot-unplugged" situation, the appropriate device driver is almost instantly notified that the PCI device has been removed. Then, the appropriate device driver will not attempt to utilize the PCI device which is no longer present in the slot or receptacle of the computer system. Thereafter, once the computer system has processing availability, the computer system can perform the administrative processing tasks associated with the removal of the previously assigned structures within the computer system for use by the previously inserted, now removed PCI device. Hence, the major processing requirements of the removal processing 300 are deferred until the computer system has sufficient processing availability so that the performance of the computer system, particularly if real-time performance, is not affected. Further, the removal of the previously assigned structures (e.g., device driver, memory space and name registry) frees up system resources for use by entities without having to reboot the system or switch between active and inactive modes.
NCLS: 345/503.000; 345/520.000; 361/118.000; 361/686.000; 361/802.000; **707/500.100; 710/002.000; 710/008.000 ; 710/017.000; 710/027.000; 710/072.000; 710/100.000; 710/104.000**

; 710/124.000; 710/300.000;
710/301.000; 710/302.000; 710/305.000
; 710/313.000

L7 ANSWER 22 OF 55 USPATFULL
PI US 6085265 20000704
TI System for handling an asynchronous interrupt a universal serial bus device
IN Kou, James Tai-Ling, Corona, CA, United States
NCL NCLM: **710/063.000**
NCLS: 370/276.000; **709/321.000; 710/001.000; 714/001.000**
AB A system and method for establishing communication between a host computer and a peripheral device. The host computer includes logic for associating an attached peripheral device with one of a particular peripheral device type, logic for associating the attached peripheral device with an instance of a software driver executable on the processor, and logic for enabling communication between the attached device and the host after a set period of time after detection of the attachment of the USB device to the at least one port. The software driver is preferably capable of supporting communication between peripheral devices of the associated particular type and the host.
CLM What is claimed is:

4. In a computer system including a host having a processor, a Universal Serial Bus (USB) host controller and a port for attaching to a USB device, the processor executing at least one applications program and one or more instances of a software driver, the software driver being capable of supporting communication between an attached USB device and the host, the USB host controller being capable of coupling to the USB device through an attachment to the port, the USB host controller including circuitry for detecting an attachment of the USB device to the port or a detachment of the USB device from the port, the improvement comprising: logic for associating an attachment of the USB device with an instance of the software driver; logic for associating the instance of the software driver with at least one application program executing on the processor, the instance of the software driver being responsive to calls from the at least one associated application program; logic for maintaining the instance of the software driver while the at least one application is executing, independent of whether the USB device remains attached to the port; and logic for terminating the instance of the software driver in response to one of a termination of the execution of the at least one application program and an acknowledgment that the **application program** has been **notified** of a **detachment** of the USB **device**.

5. The computer system of claim 4, the improvement further including: logic for maintaining a count of the executing application programs which are associated with the instance of the software driver and decrementing the count when one of the execution of an associated application terminates and an acknowledgment that the associated **application** has been **notified** of a **de**

L7 ANSWER 29 OF 55 USPATFULL
PI US 5935228 19990810
TI Method for automatically enabling peripheral devices and a storage medium for storing automatic enable program for peripheral devices
IN Shinomura, Masahiko, Machida, Japan
NCL NCLM: **710/302.000**
NCLS: **709/321.000; 713/002.000; 713/100.000**
AB A superior method for automatically enabling a PC card whereby, when a PC card that can be driven only by a DOS/Windows 3.x device driver is inserted, the PC card is enabled by that device driver, and whereby, when a PC card is inserted that can be driven by a Windows 95 native mode device driver, the PC card is enabled by that device driver. In one form the invention is directed to a method, for automatically enabling a PC card, that can be applied to an information processing system that is controlled by an operating system having either a 32-bit mode or a 16-bit mode, and that have a plurality of device drivers for both a 32-bit mode and a 16-bit mode, comprises the steps of: (a) detecting an insertion event when a PC card is inserted into the information processing system; (b) giving priority, in response to the insertion event, to the device driver of the 16-bit mode in enabling the PC card; and (c) giving chance, in response to the insertion event, to the device driver of the 32-bit mode in enabling the PC card if the PC card is not enabled by the device driver of the 16-bit mode. In another form the invention is directed to a computer implemented program operable to practice the aforementioned method.
DETD When it is ascertained that the PC card is a card that can not be enabled by the common enabler, a message to a user is displayed requesting the installation of a corresponding client device driver 53 in the system 10, i.e., that it must be copied to the HDD 17 (step 308).

When the client device driver designated 53 by a user has been installed, the **client** device driver 53 is **notified** of the **card insertion** event and the PC **card** enabling process, i.e., the allocation of system resources, is performed (step 310). As a result, the PC card is enabled in the Windows 95 native mode.
DETD If the PC card is managed by the 16-bit Card Service 61, the Socket Service 51 notifies only the 16-bit Card Service 61 of the card removal event detected by the polling (step 406). As a result, the 16-bit Card Service 61 **notifies** a corresponding **client** **device** driver 62 of the **removal** event, and upon receipt of the notice, the client device driver 62 releases the system resources allocated for the PC card and disables the PC card.
DETD When the PC card is managed by the 32-bit Card Service, the Socket Service 51 notifies only the 32-bit Card Service of the card removal event detected by the polling (step 408). As a result, the 32-bit Card Service **notifies** a corresponding **client** **device** driver 53 of the **removal** event, and upon receipt of the notice, the client device driver 53 releases the system resources allocated for the PC card and disables the PC card.
NCL NCLM: **710/302.000**
NCLS: **709/321.000; 713/002.000; 713/100.000**

L7 ANSWER 30 OF 55 USPATFULL
PI US 5887243 19990323
TI Signal processing apparatus and methods
IN Harvey, John Christopher, New York, NY, United States

L7 ANSWER 33 OF 55 USPATFULL
PI US 5748980 19980505
TI System for configuring a computer system
IN Lipe, Ralph A., Woodinville, WA, United States
Santerre, Pierre-Yves, Bellevue, WA, United States
NCL NCLM: **710/008.000**
NCLS: **710/104.000**
AB A system for configuring a devices of a computer with minimal support by
a user. Device information for the devices of the computer is collected to uniquely identify the devices and to describe the device characteristics associated with the operation of those devices with the computer. Computer resources, which support the functions of the devices
within the computer, are allocated based upon this device information. This allocation process prevents a potential conflicting use of computer
resources by the installed devices. A device driver, which enables communications between a corresponding device and the computer, is also identified and loaded for each of the devices in response to the allocation of computer resources.
DETD The operating system accommodates dynamic system events by broadcasting system messages to applications and device drivers. When an event such as the **insertion** or **removal** of a **device**
occurs, **applications** and device drivers are **notified**
. They can then take any required action or, in some cases, veto the configuration change
NCL NCLM: **710/008.000**
NCLS: **710/104.000**

L7 ANSWER 37 OF 55 USPATFULL
PI US 5623637 19970422
TI Encrypted data storage card including smartcard integrated circuit for storing an access password and encryption keys
IN Jones, Michael F., Nashua, NH, United States
Zachai, Arthur, Swampscott, MA, United States
NCL NCLM: **711/164.000**
NCLS: **710/013.000; 711/103.000; 711/115.000 ; 713/172.000; 713/185.000; 713/192.000; 713/202.000**
AB A detachable PCMCIA memory card incorporating a smartcard integrated circuit for storing a password value and logic circuitry for preventing access to information stored on the memory card unless the user of the host computer to which the memory card is connected can supply a password matching the stored password. The smartcard integrated circuit may also be used to store public and private key values used to encrypt and decrypt data stored on the card or elsewhere on the host computer or exchanged with a remote computer.
DETD The programming interface to the PCMCIA Card Services software is defined in Section 3 of the PCMCIA Standard (Release 2.01) which specifies a variety of services which are available to Client Device Drivers, as well as callback mechanisms for **notifying Client Device** Drivers of status changes. In **addition** to conventional memory operations provided by Bulk Memory Service functions, the Card Services software also provides Client Utility functions which allow client device drivers to access and manipulate the CIS stored in the memory card's attribute memory 190. Card management routines, either forming a part of the Client Device Driver or part of a special purpose application program for configuring the memory card according to the users needs, are executed on the host computer. These card management routines in turn utilize the functions provided by the PCMCIA Card Services software to implement the following two special operations which not required for conventional PCMCIA memory cards:
DETD Whenever a PCMCIA card is newly inserted into the socket of a running host computer, the Client Device Driver is **notified** by the Card Services **software** (via its **CARD**.sub.-- **INSERTION** callback function), so that it can process the card's CIS entries to identify each partition that may be password-protected. Similarly, when the host computer is first powered up and the Client Device Driver is initialized, the Client Device Driver calls Card Services functions to process the cards CIS entries to identify each partition that may be locked.
NCL NCLM: **711/164.000**
NCLS: **710/013.000; 711/103.000; 711/115.000 ; 713/172.000; 713/185.000; 713/192.000; 713/202.000**

L7 ANSWER 42 OF 55 USPATFULL
PI US 5537597 19960716
TI Method and apparatus for supporting real mode card services clients with
a protected mode card services implementation
IN Sandage, David A., Forrest Grove, OR, United States
NCL NCLM: **717/162.000**
NCLS: **709/321.000; 713/002.000**
AB PCMCIA defines a standard interface for small portable computer peripherals. Part of the PCMCIA specification defines a Card Services software layer. The current PCMCIA specification defines a Card Services
layer that provides Card Services to clients that wish to use PCMCIA cards. A Card Services Compatibility Driver is defined that ensures compatibility with real mode clients. The Card Services Compatibility Driver is a device driver or TSR that loads near the beginning of the computer system boot procedure. The Card Services Compatibility Driver simulates a full implementation of Card Services by supporting only the functions that are legal when no cards are installed in the system even though cards may be installed in the system. Later, a full protected mode implementation of Card Services is loaded into the system. The Card
Services Compatibility Driver transfers all the state information it collects during the computer system boot procedure to the full protected
mode implementation of Card Services. After receiving this information, the full protected mode implementation of Card Services takes control and services all later Card Services requests. The full protected mode implementation of Card Services also **informs** the registered **Card** Services **clients** of any **cards** **inserted** in the system by making **card** insertion callbacks.
AB PCMCIA defines a standard interface for small portable computer peripherals. Part of the PCMCIA specification defines a Card Services software layer. The current PCMCIA specification defines a Card Services
layer that provides Card Services to clients that wish to use PCMCIA cards. A Card Services Compatibility Driver is defined that ensures compatibility with real mode clients. The Card Services Compatibility Driver is a device driver or TSR that loads near the beginning of the computer system boot procedure. The Card Services Compatibility Driver simulates a full implementation of Card Services by supporting only the functions that are legal when no cards are installed in the system even though cards may be installed in the system. Later, a full protected mode implementation of Card Services is loaded into the system. The Card
Services Compatibility Driver transfers all the state information it collects during the computer system boot procedure to the full protected
mode implementation of Card Services. After receiving this information, the full protected mode implementation of Card Services takes control and services all later Card Services requests. The full protected mode implementation of Card Services also **informs** the registered **Card** Services **clients** of any **cards** **inserted** in the system by making **card** insertion callbacks.
SUMM The current PCMCIA standard defines two layers of software that a computer system uses to access and manage PCMCIA cards. The two layers

of software are called the Socket Services layer and the Card Services layer. In operation, applications which wish to use PCMCIA cards access the PCMCIA cards by using functions available from the Card Services layer. Applications which use the Card Services functions are known as Card Services "clients". A Card Services client registers with Card Services by calling a Card Services client registration function and providing some **client information** and the **address** of a **client** callback routine. **Card**

L7 ANSWER 42 OF 55 USPATFULL
PI US 5537597 19960716
TI Method and apparatus for supporting real mode card services clients with
a protected mode card services implementation
IN Sandage, David A., Forrest Grove, OR, United States
NCL NCLM: **717/162.000**
NCLS: **709/321.000; 713/002.000**
AB PCMCIA defines a standard interface for small portable computer peripherals. Part of the PCMCIA specification defines a Card Services software layer. The current PCMCIA specification defines a Card Services
layer that provides Card Services to clients that wish to use PCMCIA cards. A Card Services Compatibility Driver is defined that ensures compatibility with real mode clients. The Card Services Compatibility Driver is a device driver or TSR that loads near the beginning of the computer system boot procedure. The Card Services Compatibility Driver simulates a full implementation of Card Services by supporting only the functions that are legal when no cards are installed in the system even though cards may be installed in the system. Later, a full protected mode implementation of Card Services is loaded into the system. The Card
Services Compatibility Driver transfers all the state information it collects during the computer system boot procedure to the full protected
mode implementation of Card Services. After receiving this information, the full protected mode implementation of Card Services takes control and services all later Card Services requests. The full protected mode implementation of Card Services also **informs** the registered **Card** Services **clients** of any **cards** **inserted** in the system by making **card** insertion
callbacks.
AB PCMCIA defines a standard interface for small portable computer peripherals. Part of the PCMCIA specification defines a Card Services software layer. The current PCMCIA specification defines a Card Services
layer that provides Card Services to clients that wish to use PCMCIA cards. A Card Services Compatibility Driver is defined that ensures compatibility with real mode clients. The Card Services Compatibility Driver is a device driver or TSR that loads near the beginning of the computer system boot procedure. The Card Services Compatibility Driver simulates a full implementation of Card Services by supporting only the functions that are legal when no cards are installed in the system even though cards may be installed in the system. Later, a full protected mode implementation of Card Services is loaded into the system. The Card
Services Compatibility Driver transfers all the state information it collects during the computer system boot procedure to the full protected
mode implementation of Card Services. After receiving this information, the full protected mode implementation of Card Services takes control and services all later Card Services requests. The full protected mode implementation of Card Services also **informs** the registered **Card** Services **clients** of any **cards** **inserted** in the system by making **card** insertion
callbacks.
SUMM The current PCMCIA standard defines two layers of software that a computer system uses to access and manage PCMCIA cards. The two layers

of software are called the Socket Services layer and the Card Services layer. In operation, applications which wish to use PCMCIA cards access the PCMCIA cards by using functions available from the Card Services layer. Applications which use the Card Services functions are known as Card Services "clients". A Card Services client registers with Card Services by calling a Card Services client registration function and providing some **client information** and the **address** of a **client** callback routine. **Card** Services executes the callback routine when a PCMCIA card related event occurs.

SUMM These and other objectives are accomplished by a Card Services Compatibility Driver. The Card Services Compatibility Driver is a device driver or TSR that loads near the beginning of the computer system boot procedure. The Card Services Compatibility Driver simulates a full implementation of Card Services by supporting only the functions that are legal when no cards are installed in the system even though cards may be installed in the system. Later, a full protected mode implementation of Card Services is loaded into the system. The Card Services Compatibility Driver transfers all the state information it has collected during the computer system boot procedure to the full protected mode implementation of Card Services. After receiving this information, the full protected mode implementation of Card Services takes control and services all future Card Services requests. The full protected mode implementation of Card Services also **informs** the registered **Card** Services **clients** of any **cards inserted** in the system by making **card** insertion callbacks.

DETD However, if one or more cards are present in the system then the protected-mode implementation of Card Services may need to inform some of the registered clients of Card Services. The protected-mode implementation of Card Services first examines the list of registered clients to find the clients that wish to know about card insertions. The protected-mode implementation of Card Services then performs "callbacks" to those **clients informing** the **clients** that a **card** has been **inserted**, as stated in step 117. After **informing** all registered **clients** that wish to know about **card insertions** about all the cards present in the system, then the Card Services initialization procedure is complete.

CLM What is claimed is:

3. The method of providing Card Services functions as claimed in claim 2 wherein said step of informing said Card Services client of any PCMCIA cards installed in said computer system comprises calling a **card insertion** callback **routine** in said **Card** Services **client informing** said Card Services **client** about a PCMCIA **card insertion**.

NCL NCLM: **717/162.000**
NCLS: **709/321.000; 713/002.000**

L7 ANSWER 43 OF 55 USPATFULL
PI US 5532945 19960702
TI Power budgetting in a computer system having removable devices
IN Robinson, Kurt B., Newcastle, CA, United States
NCL NCLM: **713/321.000**
NCLS: 365/226.000; 365/227.000; **711/115.000**; **713/300.000**
AB A computer system with power budgeting for removable devices is disclosed comprising a nonvolatile memory that contains a power resource table for storing a power consumption indication for at least one resident device for the computer system. The computer system further comprises a removable device that contains a card information structure that stores a power consumption indication for the removable device. A processor executes a power management driver that allocates a power budget to the removable device according to the power resource table and the card information structure. The power management driver updates the power: resource table to indicate the power budget to the removable device.
DETD A pair of card service clients 60 and 61 are device driver programs corresponding to the application programs 50 and 51, respectively. The card service clients 60 and 61 invoke the card services 64 to access the control registers and card information structure of the removable fax-modem card 32. Card service clients 60 and 61 also invoke the card services 64 to obtain system memory and I/O mapping for the **removable** fax-modem **card** 32. The **card** services 64 **notifies** the card service **clients** 60 and 61 upon insertion and upon removal by the user of the removable fax-modem card 32.
DETD The card services 64 implement a call back interface to the card service clients 60 and 61, the logical device driver 56, and the power management driver 70. The card service clients 60 and 61, the logical device driver 56 and the power management driver 70 are all clients of the card services 64. The call back interface of the card services 64 **notifies** the appropriate **clients** when the user **inserts** a **removable device** into one of the sockets 38 or 39.
DETD A client registers for call back notification of such insertion events with a register.sub.-- client function to the card services 64. After the register **client** function, the **client** is **notified** each time a **removable device** is **inserted** into one of the sockets 38 or 39. To **notify** a **client**, the **card** services 64 invokes a clients **insertion** call back handler upon an insertion event and passes a pointer to an insertion information structure.
DETD Table 6 illustrates the format of the reset.sub.-- power information structure from the card services 64 for one embodiment. The card services 64 uses the reset.sub.-- power call back to **notify** **client device** drivers for the corresponding **removable device** that the existing Icc and Ipp voltage and current allocations are being reset to default standby levels. The default standby levels are obtained from the removable devices corresponding card information structure.
DETD At block 130, the card services **client** device driver reads the card **information** structure (CIS) of the corresponding

removable device. For example, the logical **device** driver 56 reads the card information structure memory 46 through the socket controller 26 over the system bus 34. The card information structure memory 46 indicates the peak and standby Icc and Ipp power levels for the removable flash card 30.

NCL NCLM: **713/321.000**
NCLS: 365/226.000; 365/227.000; **711/115.000; 713/300.000**

L7 ANSWER 45 OF 55 USPATFULL
PI US 5371675 19941206
TI Spreadsheet program which implements alternative range references
IN Greif, Irene, Newton Center, MA, United States
Landsman, Richard A., Hingham, MA, United States
Balaban, Robert, Cambridge, MA, United States
NCL NCLM: **707/503.000**
NCLS: 345/781.000; 345/808.000; **707/512.000**
AB A computer programmed to represent a spreadsheet that can be displayed to a user, the spreadsheet including an array of cells for storing user entered data, the programmed computer including programmed logic for designating a group of one or more cells within the spreadsheet, the group of one or more cells containing a first set of user entered data; programmed logic for creating an alternative for the group of one or more cells, said alternative containing a second set of user entered data that is different from the first set of user entered data; and programmed logic for enabling the user to select between the group of one or more cells and the alternative as the source of the data that is used in a corresponding group of cells within the spreadsheet.
DRWD Spreadsheet Services module 12 is a DLL module that interfaces directly to spreadsheet program module 10. Module 12 is implemented as a DLL so that it can be called by other modules, such as, Tools module 14. All other modules that wish to execute spreadsheet operations do so through Spreadsheet Services module 12. Other modules with callback functions resident within them may use Spreadsheet Services module 12 to register those callback functions to receive event notifications from spreadsheet program module 10. This is shown in FIG. 11 by dotted lines, representing callback **notifications, connecting** spreadsheet **program module** 10 to other modules, e.g. Tool module 14 and Event Manager module 18. In addition, Spreadsheet Services module 12 itself may register its own callback functions to provide specific functionality for other modules that do not want to receive the **notifications** directly from spreadsheet **program** module 10. Spreadsheet Services **module** 12 also contains its own **addin** shell, allowing it to communicate directly with spreadsheet program module 10.
NCL NCLM: **707/503.000**
NCLS: 345/781.000; 345/808.000; **707/512.000**

22 ANSWER 8 OF 33 USPATFULL
PI US 6055595 20000425
TI Apparatus and method for starting and terminating an application program
IN Tachibana, Yoshimi, Tokyo, Japan
Ueda, Naoki, Tokyo, Japan
NCL NCLM: **710/102.000**
NCLS: **710/104.000; 713/100.000**
AB A utility program previously registers card/**program** registration information **indicating** a correspondence relation between the names of various cards and file names of various application programs used for effecting processes for the respective cards into a registry of an OS. The utility **program** acquires information **indicating** data of **insertion**/**removal** and the name of an expansion card inserted into or removed from a card slot by referring to device information held in the registry. The utility program derives a file name of an application program corresponding to the acquired name of the expansion card by referring to the card/program registration information registered in the registry and **requesting** the OS to start the application program when the data of insertion/removal indicates the insertion.
AB A utility program previously registers card/**program** registration information **indicating** a correspondence relation between the names of various cards and file names of various application programs used for effecting processes for the respective cards into a registry of an OS. The utility **program** acquires information **indicating** data of **insertion**/**removal** and the name of an expansion card inserted into or removed from a card slot by referring to device information held in the registry. The utility program derives a file name of an application program corresponding to the acquired name of the expansion card by referring to the card/program registration information registered in the registry and **requesting** the OS to start the application program when the data of insertion/removal indicates the insertion.
SUMM According to one aspect of the present invention, there is provided an information apparatus comprising: a card slot which permits insertion/removal of an expansion card; an operating system (OS) having a registry; means for registering card/**program** registration information **indicating** a correspondence relation between names of various cards and file names of various application programs used for effecting processes for the respective cards into the registry; means for acquiring information indicating data of insertion/removal and a name of an expansion card inserted into or removed from the card slot by referring to device information held in the registry; and means for deriving a file name of an application program corresponding to the acquired name of the expansion card by referring to the card/program registration information registered in the registry and **requesting** the OS to start the application program when the data of insertion/removal indicates the insertion.

SUMM According to another aspect of the present invention, there is provided an information apparatus comprising: a card slot which permits insertion/removal of an expansion card; an operating system (OS) having a registry; means for registering card/**program** registration information **indicating** a correspondence relation between names of various cards and names of types for the

various cards and a correspondence relation between the names of the various types and file names of various application programs used for effecting processes corresponding to the names of the various types into the registry; means for acquiring information indicating data of insertion/removal and a name of an expansion card inserted into or removed from the card slot by referring to device information held in the registry; and means for deriving a name of type corresponding to the acquired name of the expansion card and deriving a file name of an application program corresponding to the derived name of type by referring to the card/program registration information registered in the registry and **requesting** the OS to start the application program when the data of insertion/removal indicates the insertion.

SUMM According to another aspect of the present invention, there is provided a program starting control method for use in an information apparatus including a card slot which permits insertion/removal of an expansion card and an operating system (OS) having a registry, the method comprising the steps of: registering card/**program** registration information **indicating** a correspondence relation between names of various cards and file names of various application programs used for effecting processes for the respective cards into the registry; acquiring information indicating data of insertion/removal and a name of an expansion card inserted into or removed from the card slot by referring to device information held in t

L22 ANSWER 23 OF 33 USPATFULL
PI US 5537597 19960716
TI Method and apparatus for supporting real mode card services clients with a protected mode card services implementation
IN Sandage, David A., Forrest Grove, OR, United States
NCL NCLM: **717/011.000**
NCLS: **709/321.000; 713/002.000**
AB PCMCIA defines a standard interface for small portable computer peripherals. Part of the PCMCIA specification defines a Card Services software layer. The current PCMCIA specification defines a Card Services layer that provides Card Services to clients that wish to use PCMCIA cards. A Card Services Compatibility Driver is defined that ensures compatibility with real mode clients. The Card Services Compatibility Driver is a device driver or TSR that loads near the beginning of the computer system boot procedure. The Card Services Compatibility Driver simulates a full implementation of Card Services by supporting only the functions that are legal when no cards are installed in the system even though cards may be installed in the system. Later, a full protected mode implementation of Card Services is loaded into the system. The Card Services Compatibility Driver transfers all the state information it collects during the computer system boot procedure to the full protected mode implementation of Card Services. After receiving this information, the full protected mode implementation of Card Services takes control and services all later Card Services **requests**. The full protected mode implementation of Card Services also **informs** the registered Card Services **clients** of any cards **inserted** in the system by making card insertion callbacks.
AB PCMCIA defines a standard interface for small portable computer peripherals. Part of the PCMCIA specification defines a Card Services software layer. The current PCMCIA specification defines a Card Services layer that provides Card Services to clients that wish to use PCMCIA cards. A Card Services Compatibility Driver is defined that ensures compatibility with real mode clients. The Card Services Compatibility Driver is a device driver or TSR that loads near the beginning of the computer system boot procedure. The Card Services Compatibility Driver simulates a full implementation of Card Services by supporting only the functions that are legal when no cards are installed in the system even though cards may be installed in the system. Later, a full protected mode implementation of Card Services is loaded into the system. The Card Services Compatibility Driver transfers all the state information it collects during the computer system boot procedure to the full protected mode implementation of Card Services. After receiving this information, the full protected mode implementation of Card Services takes control and services all later Card Services **requests**. The full protected mode implementation of Card Services also **informs** the registered Card Services **clients** of any cards **inserted** in the system by making card insertion callbacks.
SUMM These and other objectives are accomplished by a Card Services Compatibility Driver. The Card Services Compatibility Driver is a device driver or TSR that loads near the beginning of the computer system boot procedure. The Card Services Compatibility Driver simulates a full implementation of Card Services by supporting only the functions that are legal when no cards are installed in the system even though cards may be installed in the system. Later, a full protected mode implementation of Card Services is loaded into the system. The Card Services Compatibility Driver transfers all the state information it has collected during the computer system boot procedure

to the full protected mode implementation of Card Services. After receiving this information, the full protected mode implementation of Card Services takes control and services all future Card Services **requests**. The full protected mode implementation of Card Services also **informs** the registered Card Services **clients** of any cards **inserted** in the system by making card insertion callbacks.

DETD FIG. 3b illustrates how the program entities interact during the computer system boot after the Windows.TM. operating environment and the full protected mode Card Services implementation 51 have been loaded. After the computer loads the full protected mode Card Services implementation, the full protected mode Card Services implementation 51 **requests** the Card Services Compatibility Driver 50 to provide all the information that the Card Services Compatibility Driver 50 collected during the computer system boot. This information includes: the socket services data structure; a list of registered Card Services clients (including handles, callback entry points, client data, global event mask); a list of registered erase queues; a list of registered timers (with amount of time remaining on each timer); and the available resources (if appropriate for the operating environment). Note that the Card Services Compatibility Driver 50 and the full protected mode Card Services implementation 51 must perform physical/logical socket and window mapping in the same manner in order for the transition to be transparent. After obtaining the information, the full protected mode Card Services implementation 51 will take over the INT 1A entry point from the Card Services Compatibility Driver 50 and assume the role of Card Services provider to all Card Services clients. The protected mode Card Services implementation 51 then **informs** the registered Card Services **clients** about any cards present in the system by performing card insertion callbacks for all the cards present in the system.

DETD The computer proceeds to load in other TSRs and device drivers during the boot process, as stated in step 105. As each new TSR or device driver initializes itself, it may attempt to register with Card Services. The Card Services Compatibility Driver processes the registration **requests** as a normal Card Services implementation would. The Card Services Compatibility Driver, however, **informs** each registering Card Services **client** that no cards are installed in the system even if cards are installed.

DETD However, if one or more cards are present in the system then the protected-mode implementation of Card Services may need to inform some of the registered clients of Card Services. The protected-mode implementation of Card Services first examines the list of registered clients to find the clients that wish to know about card insertions. The protected-mode implementation of Card Services then performs "callbacks" to those clients **informing** the **clients** that a card has been **inserted**, as stated in step 117. After informing all registered clients that wish to know about card insertions about all the cards present in the system, then the Card Services initialization procedure is complete.

CLM What is claimed is:

3. The method of providing Card Services functions as claimed in claim 2 wherein said step of informing said Card Services client of any PCMCIA cards installed in said computer system comprises calling a card **insertion** callback **routine** in said Card Services client **informing** said Card Services **client** about a PCMCIA card **insertion**.

NCL NCLM: **717/011.000**

NCLS: **709/321.000; 713/002.000**

L22 ANSWER 29 OF 33 USPATFULL
PI US 5133075 19920721
TI Method of monitoring changes in attribute values of object in an object-oriented database
IN Risch, Tore J. M., Menlo Park, CA, United States
NCL NCLM: **707/201.000**
NCLS: **706/053.000; 710/260.000**
AB A method of monitoring objects in an interactive object-oriented database system. Any of a plurality of **client programs** can **request** monitoring of attributes of objects in the database. A record is kept of **update** transactions initiated by a **client**. When the **client** commits the **changes**, any **client** which has **requested** monitoring is **notified** of any **change** in the value of an attribute being monitored at the **request** of that **client**. The **notification** interrupts the **client** and invokes a predesignated **client** procedure. Overhead is minimized by creating partial view materialization paths and defining monitors ahead of time and by localizing the monitoring.

predesignated client procedure / call back.

AB A method of monitoring objects in an interactive object-oriented database system. Any of a plurality of **client programs** can **request** monitoring of attributes of objects in the database. A record is kept of **update** transactions initiated by a **client**. When the **client** commits the **changes**, any **client** which has **requested** monitoring is **notified** of any **change** in the value of an attribute being monitored at the **request** of that **client**. The **notification** interrupts the **client** and invokes a predesignated **client** procedure. Overhead is minimized by creating partial view materialization paths and defining monitors ahead of time and by localizing the monitoring.

SUMM A preferred embodiment of a method of monitoring an object according to the invention includes the following steps: keeping a record of any **client requests** to monitor an attribute of the object; keeping a record of any **update** transactions initiated by a **client** during an **update** session; and if that **client requests** that the transaction be committed, determining which monitored attributes may have been affected, determining whether the values of any of said attributes have changed, and, for each value which has changed, **notifying** any **client** which **requested** monitoring of that attribute.

SUMM **Notifying** a **client** preferably comprises interrupting any **task** then being performed by the **client** and invoking a predesignated **client** procedure. If the **client** is not in an interruptible state, a record is kept of **notifications** intended for that **client** until the **client** enters an interruptible state or **requests notification**.

SUMM Keeping the record of update transactions imposes very little overhead on the database system because the record is small and can be kept in main memory. The record can be cleared after determining whether the values of any attributes have changed. This determination may be made when updates are committed or any time a **client requests notification** of any changes. In the latter

event, a record of the changes is preferably kept for later notification of other clients for which attributes are being monitored.

SUMM Certain high-overhead tasks can be performed before an update session is commenced. One such task consists of optimizing a partial view materialization path for each attribute of a view of an object. This task is preferably carried out during creation of the view itself. Later, if monitoring of the value of an attribute of that view is requested, the optimized path for that attribute is used, thereby imposing less overhead on the system than would be imposed if the entire view had to be materialized in order to monitor the value of that one attribute.

DETD A method of monitoring a database object according to the invention includes keeping a record of any client request to monitor an attribute of the object, keeping a record of update transactions, and notifying the requesting client of any change in the value of the monitored attribute by invoking a predetermined client procedure when the update transactions are committed. The invention thus provides an efficient means for client programs to initiate monitor requests and receive notifications of changes in monitored attributes of database objects while imposing only minimal overhead on the database system.

DETD In a preferred embodiment, a method of monitoring an object in a database in response to a request from any of a plurality of client programs comprises keeping a record of any client requests to monitor an attribute of the object; during a database update session, keeping a record of any database update transactions initiated by a client ; and, if a client which has initiated an update transaction requests that the transaction be committed, determining which monitored attributes may have been affected by said transaction, determining whether the values of any of said attributes have changed, and for each value which has changed notifying any client which requested monitoring of that attribute.

DETD The record of database update transactions is preferably kept by creating and updating a "Function Change" table during an "Update Session" procedure as illustrated in FIG. 2. The procedure is begun (block 201) by a client initiating an update session. The system creates the Function Change table (block 202) and then proceeds with an update task as directed by the client (block 203). If the task included an extensional function update call (block 204), the function which was called is entered in the Function Change table (block 205); if not, no entry is made in the table. The client decides whether to commit any updates made thus far (block 206), and if the client requests that such updates be committed a "Check Monitors" procedure is called (block 207).

DETD Determining which monitored attributes may have been affected and whether the values of any of those attributes have changed is preferably accomplished by means of the Check Monitors procedure as illustrated in FIG. 3. As described above, the procedure is initiated (block 301) by a procedure call generated during an update session (the procedure may also be called directly by a client as will be discussed in more detail in a subsequent paragraph). Any update function call which might have resulted in a change to a monitored attribute value is detected (block 302) by reference to the Function Change table and to a "Function Dependence" table (to be described hereafter). An updated value for each such attribute is computed (block 303). Whether there has been any change in such an a

22 ANSWER 31 OF 33 USPATFULL
PI US 5109486 19920428
TI Distributed computer system with network and resource status monitoring
IN Seymour, Leslie G., Barrington, IL, United States
NCL NCLM: **709/224.000**
NCLS: **709/220.000**
AB A **requesting** manager sends a message to its local configuration processor **requesting** notice of status changes in a general or particular, resource or node. The configuration manager establishes a record in its status monitoring file containing the information from the **requesting process'** message. The configuration manager receives messages: from remote configuration managers or local resource managers regarding resource status changes; and from its local network interface module regarding node changes. The configuration manager then compares these status change notifications with the records in its status monitoring file. If there is a match, the configuration manager sends a message to the **requesting process reporting** the status **change**. The status **changes** that are **reported** may be limited by the **requesting process** by specifying a particular status change, a particular resource, or other option.
AB A **requesting** manager sends a message to its local configuration processor **requesting** notice of status changes in a general or particular, resource or node. The configuration manager establishes a record in its status monitoring file containing the information from the **requesting process'** message. The configuration manager receives messages: from remote configuration managers or local resource managers regarding resource status changes; and from its local network interface module regarding node changes. The configuration manager then compares these status change notifications with the records in its status monitoring file. If there is a match, the configuration manager sends a message to the **requesting process reporting** the status **change**. The status **changes** that are **reported** may be limited by the **requesting process** by specifying a particular status change, a particular resource, or other option.
SUMM A particular embodiment of the present invention comprises a distributed computer system having a plurality of nodes with each node comprising a network interface module and a configuration manager. In operation, the configuration manager is notified by local resource managers on its node of the status situations of **processes** to which they desire to be kept apprised. The configuration manager of the node interacts with configuration managers of other nodes to **inform** the **requesting processes** of the statuses **requested**. The network interface modules also communicate with the configuration managers to inform the configuration managers when the status of any node changes. This information is used by the configuration manager to locate the remote nodes of which it was maintaining a status check. The **requesting processes** are then **notified** of the change in status.

DETD In operation, a **process** may be created which requires access to a particular resource. However, before accessing the resource, the **process** needs to know the status of the resource (i.e. is it present on the system, is it busy, is it available, etc.). Since the

process can not make this determination itself, a request is sent to the local configuration manager. The configuration manager will then determine the status of the resource and report back to the requesting process. This status determination is made by contacting the remote configuration manager for the node containing the resource. User transparent distributed services are provided by the use of the configuration manager for the described network monitoring process.

DETD In a more detailed explanation of the operation, there are four different types of messages used in the present invention. These are designated SNREQ (Status Notification REQuest), RSREP (Resource Status REPort), SNREP (Status Notification REPort), and NSR (Node Status Report). The SNREQ is sent by a process to configuration manager 204. The SNREQ message is in the form:

DETD If message (1) is sent, as it appears above, each "status" message sent to configuration manager 204 is reported to the requesting process. There are three ways to restrict the status messages reported to the requesting process. These restrictions are implemented by adding an option to the message. One of the options is set forth below in message (2).

DETD Following NODE NAME segment 277 is a STATUS CODE segment 278. Segment 278 is used if the message specified that the requestor was only interested in certain status changes of the resource. If the STATUS CODE is unspecified in the message, then all status changes are reported to the requesting process.

DETD Following RESOURCE CONNECTOR segment 279 is a REQUEST CONNECTOR segment 280. Segment 280 is the identification of the requesting process. This informs the configuration manager who to contact when the monitored status change occurs. Finally, a HOST ID segment 281 is provided to identify the ID of the node; the status of which is requested to be monitored. Since the HOST ID is made up of a combination of CPU and node ID's, this number can also be used to identify the particular CPU on which the process resides.

DETD The second type of message is transmitted from a resource manager to the configuration manager and is labeled RSREP (Resource Status REPort). This message is sent to the local configuration manager whenever the status of a resource on that manager's node changes. The local configuration manager then may transmit this status change to the remote configuration managers. The configuration managers check their lists of status requests for matches to the RSREP. If the report matches a request, the requesting process is notified of the status change by a SNREP (Status Notification REPort) message.

DETD Likewise, if the configuration manager receives a message that a new node has been connected, the configuration manager checks its status monitoring file to identify any requests dealing with the new node or with the system in general. In the prior instance, the configuration manager will inform the requesting process of the status change. In the later case, the configuration manager will automatically establish new status request records in the file for resources on the new node and may inform the requesting processes.

DETD Referring now to the flow charts of FIGS. 10-14, a process embodying the present invention is illustrated. Flow charts 10-14 relate to various portions of the code listing contain in Appendix A. In FIG. 10, a process of setting and canceling status notification requests is illustrated. The routine commences at decision block 300 where the process checks to see if there is a resource connector in a request message received by the configuration manager, line 64

o

L22 ANSWER 5 OF 33 USPATFULL
PI US 6163795 20001219
TI Server for notifying items of interest and delivering locally accessed video in a WAN to client stations on demand
IN Kikinis, Dan, Saratoga, CA, United States
NCL NCLM: **709/203.000**
NCLS: **709/219.000; 725/087.000**
AB A service for delivering locally accessed video to client stations on demand includes a plurality of client stations used by a client and adapted to receive and play digital video entities. A plurality of file servers remote from the client stations are connected on a wide area network (WAN), the file servers also have connections to video input apparatus and receive and store digital video entities including newly downloaded locally accessed video entities and are also connected to at least one of the plurality of client stations by a digital communication link. The file servers automatically transmit in bursts the newly downloaded locally accessed video entities to each of the plurality of file servers on the network through the WAN **connections**, store details of **connected clients'** interests provided by the **client**, **notify** the **connected clients** of new video entities available conforming to the stored details of the clients interests, and transmit selected video entities to at least one of the client stations on demand. The file servers in most embodiments have all of the functions of the other file servers on the WAN including a local client base. In some embodiments the the file servers are located in or near population centers in a global network and the locally accessed video entities are news programs from in or near the population centers related to the file servers.

gen'l concept

AB A service for delivering locally accessed video to client stations on demand includes a plurality of client stations used by a client and adapted to receive and play digital video entities. A plurality of file servers remote from the client stations are connected on a wide area network (WAN), the file servers also have connections to video input apparatus and receive and store digital video entities including newly downloaded locally accessed video entities and are also connected to at least one of the plurality of client stations by a digital communication link. The file servers automatically transmit in bursts the newly downloaded locally accessed video entities to each of the plurality of file servers on the network through the WAN **connections**, store details of **connected clients'** interests provided by the **client**, **notify** the **connected clients** of new video entities available conforming to the stored details of the clients interests, and transmit selected video entities to at least one of the client stations on demand. The file servers in most embodiments have all of the functions of the other file servers on the WAN including a local client base. In some embodiments the the file servers are located in or near population centers in a global network and the locally accessed video entities are news programs from in or near the population centers related to the file servers.

SUMM In a preferred embodiment, a video jukebox system is provided. The system comprises a plurality of computerized file server stations interconnected in a network by first data links, with individual ones of the file server stations also connected to a video input apparatus, such as a VCR device or other device capable of providing video input to the connected file server. There is a plurality of client stations connected by second data links to individual ones of the file server stations. Each file server station with a video input apparatus

accepts video clippings from the video input apparatus, stores the clippings in a database, and shares the video clippings with other file servers in the network over the first data links, and wherein each file server **notifies connected clients** of new video clippings available, and downloads video clippings to clients on demand.

DETD FIG. 2 is a flow diagram **indicating** the functionality of control **routines** 16 executed on the file servers in the global network of file servers. There are at least three different inputs that may be expected at a local file server: (1) an operator-triggered and annotated local video input, which will be in most cases a new video clipping to add to the database (2) a compressed clipping received from another server, which will be in most cases a new clipping to be stored and shared with local subscribers as well, and (3) a **request** from a local subscriber, which in most cases will be a **request** for downloading one or more clippings.

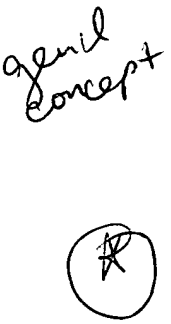
DETD When **client** 225's request for a movie or clipping arrives at account manager 229, the request is processed. **Client** 225's account is charged for the particular movie time or clipping **requested**, and the **updated** account is then available for whatever billing cycle is in use. The account manager also associates the material (movie or clipping) requested with the resource location, and initiates the **process** of sending the data to **client** 225's site.

CLM What is claimed is:

1. A video delivery system, comprising: a plurality of client stations used by a client and adapted to receive and play digital video entities; and a plurality of file servers remote from the client stations, the file servers connected on a wide area network (WAN), the file servers also having connections to video input apparatus for receiving and storing digital video entities including newly downloaded locally-accessed video entities, and connected to at least one of the plurality of client stations by a digital communication link; wherein the file servers automatically transmit in bursts the newly downloaded locally-accessed video entities to each of the plurality of file servers on the network through the WAN **connections**, store details of **connected clients** interests provided by the **client**, **notifies** the **connected clients** of new video entities available conforming to the stored details of the clients interests, and transmits selected video entities to at least one of the client stations on demand.

9. A method for providing video entities to clients, comprising steps of: (a) storing details of client interests in a database on a file server; (b) loading video entities including locally accessed video entities to the database of the file server having a connection to a plurality of file servers over a wide area network (WAN); (c) transmitting the video entities automatically including the locally accessed video entities to the other file servers on the WAN; (d) **no**

L22 ANSWER 14 OF 33 USPATFULL
PI US 5826253 19981020
TI Database system with methodology for **notifying** **clients** of any **additions**, deletions, or modifications occurring at the database server which affect validity of a range of data records cached in local memory buffers of clients
IN Bredenberg, David, Acton, MA, United States
NCL NCLM: **707/002.000**
NCLS: **707/001.000; 707/007.000; 707/008.000; 707/010.000; 707/100.000 ; 711/100.000; 711/103.000**
AB **Client**/server system and methods are described for providing a "cache range" to database **clients**. When one or more records in a cache range of a **client** change, a server in **connection** with the **client** sends a **notification** that the cache range has **changed**. Instead of the **client** taking a lock out for **updating** a record, the **client** simply **indicates** to the server which records it is interested in (e.g., via registering an event **alerter request**), whereupon the server manages its resources as necessary to **notify** the **client** of a **change** in one of the records which is of interest to the **client**. The server can simply take out "interest" locks on the corresponding records; these are not locks which exclude the resource from other **clients**. Other **clients** are not prevented from accessing this resource (i.e., the records which are of interest). The interest lock is employed in conjunction with the registered event **alerter** for **notifying** the **client** when a range of records of interest is **updated** by another **client**. In this manner, the server undertakes action to **indicate** to the **client** when the **client** local buffer might be stale, thereby providing improved local record buffer management in a **client**/server database context.
TI Database system with methodology for **notifying** **clients** of any **additions**, deletions, or modifications occurring at the database server which affect validity of a range of data records cached in local memory buffers of clients
AB **Client**/server system and methods are described for providing a "cache range" to database **clients**. When one or more records in a cache range of a **client** change, a server in **connection** with the **client** sends a **notification** that the cache range has **changed**. Instead of the **client** taking a lock out for **updating** a record, the **client** simply **indicates** to the server which records it is interested in (e.g., via registering an event **alerter request**), whereupon the server manages its resources as necessary to **notify** the **client** of a **change** in one of the records which is of interest to the **client**. The server can simply take out "interest" locks on the corresponding records; these are not locks which exclude the resource from other **clients**. Other **clients** are not prevented from accessing this resource (i.e., the records which are of interest). The interest lock is employed in conjunction with the registered event **alerter** for **notifying** the **client** when a range of records of interest is **updated** by another **client**. In this manner, the server undertakes action to **indicate** to the **client** when the **client** local buffer might be stale,



thereby providing improved local record buffer management in a **client**/server database context.

SUMM Employing an event **alerter**-based cache range, the foregoing **client**/server approach to **updating** records can be reversed as follows. Instead of the **client** taking a lock out for **updating** a record, the **client** simply **indicates** to the server which records it is interested in (e.g., via registering an event **alerter request**), whereupon the server manages its resources as necessary to **notify** the **client** of a change in one of the records which is of interest to the **client**. In effect, instead of a **client**/server approach to **updating** records, a server/**client** approach is adopted.

SUMM The **client indicates** to the server which record it is interested in when the initial **request** for the records is made. Thus at the time of submitting this **request**, the **client** can simply also **indicate** in the same **request** that these are records which the **client** desires to know if any changes are posted. Accordingly, the server can simply take out "interest" locks on the corresponding records; these are not locks which exclude the resource from other **clients**. Other **clients** are not prevented from accessing this resource (i.e., the records which are of interest). Instead, each interest lock s

20 ANSWER 1 OF 31 USPATFULL
PI US 6266716 B1 20010724
TI Method and system for controlling data acquisition over an information bus
IN Wilson, Douglass J., Boston, MA, United States
Colan, Mark T., Medford, MA, United States
NCL NCLM: **710/033.000**
NCLS: 709/213.000; 709/216.000; 709/302.000; 709/303.000
AB A group of protocols is described that establish an **information** bus. The protocols allow various **applications** and components to **plug** into the information bus. As a member of the bus, each **application** or component can exchange **information** with any other **application** or component in a structured way. The information bus is especially useful in interconnecting Java beans and applets in a Java virtual machine and in a distributive computer environment. An apparatus is disclosed that is utilized to produce data from an **application** to an **information** bus for sharing the data with other **applications** utilizing the **information** bus. The data producing apparatus comprises a data element building logic, which builds data elements containing the data within the **application**, a data **notification** logic, which **notifies** the information bus of the availability of the data element, and a data element transfer logic, which transfers data element from one application to another. An event listing logic can also be included that is utilized to listen for data element requests from **applications** utilizing the **information** bus. The data element builder logic may be configured to change the data within a data element and the data notification logic may be configured to announce the data change across the information bus to at least some of the applications. The data element builder logic specifies the data name using either a property or a parameter. The data element builder logic also provides a view associated with the data item where the data item is a java object. The data element builder logic may also **remove** data elements for access while the data notification logic notifies the information bus of the **removal** of the particular data elements.
AB A group of protocols is described that establish an **information** bus. The protocols allow various **applications** and components to **plug** into the information bus. As a member of the bus, each **application** or component can exchange **information** with any other **application** or component in a structured way. The information bus is especially useful in interconnecting Java beans and applets in a Java virtual machine and in a distributive computer environment. An apparatus is disclosed that is utilized to produce data from an **application** to an **information** bus for sharing the data with other **applications** utilizing the **information** bus. The data producing apparatus comprises a data element building logic, which builds data elements containing the data within the **application**, a data **notification** logic, which **notifies** the information bus of the availability of the data element, and a data element transfer logic, which transfers data element from one application to another. An event listing logic can also be included that is utilized to listen for data element requests from **applications** utilizing the **information** bus. The data element builder logic may be configured to change the data within a data element and the data notification logic may be configured to announce the data change across the information bus to

at least some of the applications. The data element builder logic specifies the data name using either a property or a parameter. The data element builder logic also provides a view associated with the data item where the data item is a java object. The data element builder logic may also **remove** data elements for access while the data notification logic notifies the information bus of the **removal** of the particular data elements.

NCL NCLM: **710/033.000**
NCLS: 709/213.000; 709/216.000; 709/302.000; 709/303.000

L20 ANSWER 2 OF 31 USPATFULL
PI US 6256739 B1 20010703
TI Method and apparatus to determine user identity and limit access to a communications network
IN Skopp, Peter, New York, NY, United States
Vitale, Benjamin F., New York, NY, United States
Marur, Vinod R., Ridgefield, CT, United States
Tse, Clifford S.C., New York, NY, United States
Dulai, Dharmender S., New York, NY, United States
NCL NCLM: **713/201.000**
NCLS: 709/229.000
AB A method and apparatus to determine user identity and limit access to a communications network. A first message containing user identity **information** is received from a **client** computer in accordance with a first protocol. A first network **address** is determined from the first message. A second message containing an information request is also received from the client in accordance with a second protocol, and a second network **address** is determined from the second message. The requesting user identity is then determined based on the first network **address**, the user identity information and the second network **address**. Based on the requesting user identity, it can be decided whether to grant the information request. If access is granted, the requested information is retrieved using the communications network.
AB A method and apparatus to determine user identity and limit access to a communications network. A first message containing user identity **information** is received from a **client** computer in accordance with a first protocol. A first network **address** is determined from the first message. A second message containing an information request is also received from the client in accordance with a second protocol, and a second network **address** is determined from the second message. The requesting user identity is then determined based on the first network **address**, the user identity information and the second network **address**. Based on the requesting user identity, it can be decided whether to grant the information request. If access is granted, the requested information is retrieved using the communications network.
NCL NCLM: **713/201.000**
NCLS: 709/229.000

L22 ANSWER 3 OF 20 USPATFULL
PI US 2001002473 A1 20010531
TI Dynamic lookup service in a distributed system
IN Waldo, James H., Dracut, MA, United States
Wollrath, Ann M., Groton, MA, United States
Scheifler, Robert W., Somerville, MA, United States
Arnold, Kenneth C.R.C., Lexington, MA, United States
NCL NCLM: **709/229.000**
NCLS: **709/220.000; 709/227.000; 709/225.000**
AB An improved lookup service is provided that allows for the dynamic **addition** and deletion of services. This lookup service allows for the **addition** and deletion of services automatically without user intervention. As a result, clients of the lookup service may continue using the lookup service and its associated services while the updates occur. **Additionally**, the lookup service provides a notification mechanism that can be used by **clients** to receive a **notification** when the lookup service is **updated**. By receiving such a **notification**, **clients** can avoid attempting to access a service that is no longer available and can make use of new services as soon as they are **added** to the lookup service.
AB An improved lookup service is provided that allows for the dynamic **addition** and deletion of services. This lookup service allows for the **addition** and deletion of services automatically without user intervention. As a result, clients of the lookup service may continue using the lookup service and its associated services while the updates occur. **Additionally**, the lookup service provides a notification mechanism that can be used by **clients** to receive a **notification** when the lookup service is **updated**. By receiving such a **notification**, **clients** can avoid attempting to access a service that is no longer available and can make use of new services as soon as they are **added** to the lookup service.
SUMM [0031] In accordance with methods consistent with the present invention, a method is provided in a data processing system having a lookup service with associated services. This method receives a **request** by the lookup service for **notification** when the lookup service is updated, determines when the lookup service is updated, and generates a notification when it is determined that the lookup service is updated.

SUMM [0034] In accordance with systems consistent with the present invention, a data processing system containing a memory and a p

L22 ANSWER 9 OF 20 USPATFULL
PI US 6038625 20000314
TI Method and system for providing a device identification mechanism within a consumer audio/video network
IN Ogino, Hiroshi, Sunnyvale, CA, United States
Zou, Feng (Frank), Milpitas, CA, United States
NCL NCLM: **710/104.000**
NCLS: **709/328.000; 710/008.000; 710/009.000; 710/010.000; 712/001.000 ; 712/208.000**
AB A method and system for providing a device identification mechanism within a consumer electronics based audio/video network. Several consumer electronics products, e.g., television, VCR, tuner, set-top box (e.g., intelligent receiver/decoder, IRD), DVTRs, PCs, DVD players (digital video disk), etc., can be **coupled** within the network to communicate together via a standard bus (e.g., IEEE 1394 serial communication bus). In one embodiment, the HAVI network offers unique advantages consumer electronic vendors because the architecture offers for the home network many of the advantages of existing computer system networks. Specifically, interconnected devices can share resources and provide open, well defined APIs that allow ease of development for third party developers. The present invention provides a mechanism whereby a global unique identifier (GUID) is associated with each device of the HAVI network. A low level driver constructs a GUID list of each device on the HAVI network. The order of the GUID entries in the GUID list (e.g., the index) matches the physical identifiers assigned to the devices by the 1394 serial bus. Although the physical identifiers can change on bus reset, the GUID values are constant and are used for device communication. Speed map and topology map information is maintained based on the physical identifier information and therefore translations between GUIDs and physical identifiers are efficiently performed by the present invention when referencing speed map and topology **information** for an **application**.
AB A method and system for providing a device identification mechanism within a consumer electronics based audio/video network. Several consumer electronics products, e.g., television, VCR, tuner, set-top box (e.g., intelligent receiver/decoder, IRD), DVTRs, PCs, DVD players (digital video disk), etc., can be **coupled** within the network to communicate together via a standard bus (e.g., IEEE 1394 serial communication bus). In one embodiment, the HAVI network offers unique advantages consumer electronic vendors because the architecture offers for the home network many of the advantages of existing computer system networks. Specifically, interconnected devices can share resources and provide open, well defined APIs that allow ease of development for third party developers. The present invention provides a mechanism whereby a global unique identifier (GUID) is associated with each device of the HAVI network. A low level driver constructs a GUID list of each device on the HAVI network. The order of the GUID entries in the GUID list (e.g., the index) matches the physical identifiers assigned to the devices by the 1394 serial bus. Although the physical identifiers can change on bus reset, the GUID values are constant and are used for device communication. Speed map and topology map information is maintained based on the physical identifier information and therefore translations between GUIDs and physical identifiers are efficiently performed by the present invention when referencing speed map and topology **information** for an **application**.
SUMM A list of available devices in the network is **information**

that network-based applications generally **request** periodically. The present invention generates and maintains a GUID list of all devices of the network and orders the GUIDs of the GUID list by their respective physical identifier values. In this manner, the index of a particular GUID within the GUID list is its physical identifier and this index can be obtained and then used to access data from the speed map and topology map data structures which are constructed and maintained with respect to physical identifier values. For instance, if the speed between devicex (GUID1) and devicey (GUID2) is needed, the present invention locates the index values, index1 and index2, for GUID1 and GUID2 using the GUID list. Index1 and index2 are then used to access the speed map data structure and the desired speed data is returned to higher level applications. Alternatively, the entire speed map or topology map can be presented to the higher level application along with the current GUID list and the application can perform the indexing for the desired data.

DETD One of the advanced features the 1394 bus provides to the HAVI architecture is its support for dynamic device actions such as hot plugging (device insertion or power up) and unplugging (device removal or power down). To fully support this to the user level, high level software clients need to be aware of these environment changes and the present invention provides this information. The CMM 250 works with the Event Manager 214 to detect and announce these dynamic bus changes u